IBM VSE/Enterprise Systems Architecture

System Macros Reference

Version 2 Release 4

SC33-6716-00

IBM VSE/Enterprise Systems Architecture

System Macros Reference

Version 2 Release 4

SC33-6716-00

Note!

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

Second Edition (June 1999)

This edition applies to Version 2 Release 4 of IBM Virtual Storage Extended/Enterprise Systems Architecture (VSE/ESA), Program Number 5690-VSE, and to all subsequent releases and modifications until otherwise indicated in new editions.

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

A form for readers' comments is provided at the back of this publication. If the form has been removed, address your comments to:

IBM Corporation
Attn: Dept ECJ-BP/003D
6300 Diagonal Highway
Boulder, CO 80301
U.S.A.

or to:

IBM Deutschland Entwicklung GmbH
Department 3248
Schoenaicher Strasse 220
D-71032 Boeblingen
Federal Republic of Germany

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

© **Copyright International Business Machines Corporation 1990, 1999. All rights reserved.**
Note to U.S. Government Users — Documentation related to restricted rights — Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

Contents

Notices . . . xiii
Programming Interface Information . . . xiii
Trademarks and Service Marks . . . xiv

About This Book . . . xv
Who Should Use This Book . . . xv
Where to Find More Information . . . xv

Summary of Changes . . . xvii

Chapter 1. Using the Macros . . . 1
Selecting the Macro Level . . . 1
Addressing Mode and the Macros . . . 2
Address Space Control (ASC) Mode . . . 2
Using X-Macros . . . 3
Passing Parameters in AR Mode . . . 3
Register Usage . . . 4
Macro Notation . . . 4
- Positional Operands . . . 5
- Keyword Operands . . . 5
- Mixed Format . . . 5
- Comments in Macros . . . 5
- Understanding Syntax Diagrams . . . 5
- Register Notation . . . 8

Chapter 2. Macro Descriptions . . . 11
ALESERV (Access List Entry) Macro . . . 11
ALESERV ADD (Add Access List Entry) Macro . . . 12
- Return Codes in Register 15 . . . 13

ALESERV DELETE (Delete Access List Entry) Macro . . . 14
- Return Codes in Register 15 . . . 14

ALESERV EXTRACT (Find a STOKEN) Macro . . . 15
- Return Codes in Register 15 . . . 16

ALESERV SEARCH (Search Access List Entry) Macro . . . 17
- Return Codes in Register 15 . . . 18

AMODESW CALL (Addressing Mode Switch) Macro . . . 19
AMODESW QRY (Query Addressing Mode) Macro . . . 20
AMODESW RETURN (Return from Subroutine) Macro . . . 20
AMODESW SET (Set Addressing Mode) Macro . . . 21
ASPL (Assign Parameter List) Macro . . . 22
ASSIGN (Assign I/O Device) Macro . . . 23
- Return Codes in Register 15 . . . 23

ATTACH (Attach a Task) Macro . . . 24
AVRLIST (Map GETVCE) Macro . . . 27
CALL (Call a Program) Macro . . . 28
CALL CSRPxxx (Call Cell Pool Services) Macro . . . 30
- Control Parameters . . . 31
- Programming Requirements . . . 31
- Register Information . . . 31

CALL CSRPBLD (Build A Cell Pool And Initialize An Anchor) . . . 32

© Copyright IBM Corp. 1990, 1999

Return Codes in Register 15 . . . 33
CALL CSRPEXP (Expand A Cell Pool) . . . 34
Return Codes in Register 15 . . . 35
CALL CSRPCON (Connect Cell Storage to an Extent) . . . 36
Return Codes in Register 15 . . . 36
CALL CSRPACT (Activate Previously Connected Storage) . . . 38
Return Codes in Register 15 . . . 38
CALL CSRPDAC (Deactivate an Extent) . . . 39
Return Codes in Register 15 . . . 40
CALL CSRPDIS (Disconnect the Cell Storage for an Extent) . . . 41
Return Codes in Register 15 . . . 41
CALL CSRPGET (Allocate a Cell from a Cell Pool) . . . 42
Return Codes in Register 15 . . . 43
CALL CSRPRGT (Allocate a Cell from a Cell Pool - Register Interface) . . . 44
Input Register Information . . . 44
Output Register Information . . . 44
Return Codes in Register 15 . . . 45
CALL CSRPFRE (Return a Cell to a Cell Pool) . . . 45
Return Codes in Register 15 . . . 46
CALL CSRPRFR (Return a Cell to a Cell Pool - Register Interface) . . . 47
Input Register Information . . . 47
Output Register Information . . . 47
Return Codes in Register 15 . . . 48
CALL CSRPQPL (Query the Cell Pool) . . . 49
Return Codes in Register 15 . . . 50
CALL CSRPQEX (Query a Cell Pool Extent) . . . 51
Return Codes in Register 15 . . . 52
CALL CSRPQCL (Query a Cell) . . . 53
Return Codes in Register 15 . . . 54
CANCEL (Cancel Task) Macro . . . 55
CCB (Command Control Block Definition) Macro . . . 56
Format of the CCB . . . 57
CCB Communication Bytes . . . 58
CDDELETE (Delete Loaded Phase) Macro . . . 62
Return Codes in Register 15 . . . 62
CDLOAD (Control-Directory Load) Macro . . . 63
Return Codes in Register 15 . . . 64
CDMOD (Card I/O Module Definition) Macro . . . 65
Standard CDMOD Names . . . 67
Subset/Superset CDMOD Names . . . 67
CHAP (Change Priority) Macro . . . 68
CHECK (Check I/O Completion) Macro . . . 69
CHKPT (Checkpoint Request) Macro . . . 72
CLOSE and CLOSER (Close a File) Macro . . . 75
CNTRL (Control Device) Macro . . . 76
CKD-Disk Devices . . . 77
Magnetic Tape Units . . . 77
Printers – Any Type . . . 77
PRT1 Printers – Including IBM 4248 in Native Mode . . . 78
Card I/O Devices . . . 79
IBM 3881 Optical Mark Reader . . . 79
IBM 3886 Optical Character Reader . . . 79
COMRG (Communication Region Access) Macro . . . 79
CPCLOSE (Control Program File Close) Macro . . . 80

Return Codes in Register 15 . . . 80
CSRCMPSC (Compression/Expansion) Macro . . . 81
Return Codes . . . 81
CSRYCMPS (Map Compression Control Block) Macro . . . 82
DCTENTRY (Map GETVCE) Macro . . . 82
DEQ (Dequeue Resource) Macro . . . 83
DETACH (Detach Task) Macro . . . 84
DFR (Define Font Record) Macro . . . 85
DIMOD (Device-Independent I/O Module Definition) Macro . . . 87
Standard DIMOD Names . . . 88
Subset/Superset DIMOD Names . . . 88
DISEN (Disengage Document Reader) Macro . . . 89
DLINT (Define Line Type) Macro . . . 90
Line-Information Specifications . . . 90
Field-Information Specifications . . . 90
DOM (Delete Operator Message) Macro . . . 93
Return Codes in Register 15 . . . 93
Cancel Codes . . . 93
DRMOD (Document Read Module Definition) Macro . . . 94
Standard DRMOD Names . . . 94
DSPLY (Display Document Field) Macro . . . 95
DSPSERV (Data Space) Macro . . . 96
DSPSERV CREATE (Create Data Space) Macro . . . 97
Return Codes in Register 15 (and Reason Codes in Register 0) . . . 101
Data Space Naming Conventions . . . 101
DSPSERV DELETE (Delete Data Space) Macro . . . 102
DSPSERV EXTEND (Extend Data Space) Macro . . . 104
Return Codes in Register 15 (and Reason Codes in Register 0) . . . 105
DSPSERV RELEASE (Release Data Space) Macro . . . 106
DTFCD (Define the File for Card I/O) Macro . . . 108
DTFCN (Define the File for Console I/O) Macro . . . 115
DTFDA (Define the File for Direct Access) Macro . . . 117
DTFDI (Define the File for Device Independence) Macro . . . 123
DTFDR (Define the File for Document Reader) Macro . . . 127
DTFDU (Define the File for Diskette Unit I/O) Macro . . . 130
DTFIS (Define the File for Indexed Sequential Access) Macro . . . 135
DTFMR (Define the File for Magnetic Reader Input) Macro . . . 145
DTFMT (Define the File for Magnetic Tape I/O) Macro . . . 147
DTFOR (Define the File for Optical Reader Input) Macro . . . 154
DTFPH (Define the File for Physical I/O) Macro . . . 160
DTFPR (Define the File for Printer) Macro . . . 164
DTFSD (Define the File for Sequential Disk I/O) Macro . . . 169
DTL (Define the Lock) Macro . . . 177
DUMODFx (Diskette Unit I/O Module Definition) Macro . . . 179
Standard DUMOD Names . . . 179
Subset/Superset DUMOD Names . . . 180
DUMP (Dump Request) Macro . . . 180
ENDFL (End File Load Mode) Macro . . . 182
ENQ (Enqueue a Task) Macro . . . 183
EOJ (End of Job) Macro . . . 184
ERET (Error-Handling Return) Macro . . . 184
ESETL (End Set Limit) Macro . . . 185
EXCP (Execute Channel Program) Macro . . . 186
EXIT (Return from Exit Routine) Macro . . . 187

STXIT Macro Issued With AMODE 24 . . . 188
STXIT Macro Issued With AMODE ANY . . . 188
EXTRACT (Extract Control Information) Macro . . . 189
Return Codes in Register 15 . . . 191
FCEPGOUT (Force Page Out) Macro . . . 191
Exceptional Conditions . . . 193
Return Codes in Register 15 . . . 193
FEOV (Force End of Volume) Macro . . . 193
FEOVD (Force End of Volume for Disk) Macro . . . 194
FETCH (Fetch a Phase) Macro . . . 195
Return Codes in Register 15 . . . 197
FREE (Free Disk Area) Macro . . . 198
FREEVIS (Free Virtual Storage) Macro . . . 199
Format 1: Freeing Storage from the Partition FREEVIS Area . . . 199
Format 2: Freeing Storage from the Space FREEVIS Area . . . 199
Format 3: Freeing Storage from the System FREEVIS Area . . . 199
GENDTL (Generate the DTL Block) Macro . . . 201
GENIORB (Generate an IORB) Macro . . . 204
GENL (Generate Directory List) Macro . . . 205
GET (Get a Record) Macro . . . 207
GETIME (Get the Time) Macro . . . 208
GETSYMB (Get Symbolic Parameter) Macro . . . 210
GETVCE (Get Volume Characteristics) Macro . . . 211
GETVCE Output . . . 214
Return Codes in Register 15 . . . 214
AVRLIST and DCTENTRY . . . 215
GETVIS (Get Virtual Storage) Macro . . . 217
Format 1: Obtaining Storage from the Partition GETVIS Area . . . 217
Format 2: Obtaining Storage from the Space GETVIS Area . . . 217
Format 3: Obtaining Storage from the System GETVIS Area . . . 217
IJBPUB (IJBPUB DSECT) Macro . . . 222
IJJLBSER (LBSERV DSECT) Macro . . . 222
IORB (I/O Request Block Definition) Macro . . . 224
ISMOD (Indexed Sequential I/O Module Definition) Macro . . . 226
Standard ISMOD Names . . . 228
Subset/Superset ISMOD Names . . . 229
JDUMP (Job Dump Request) Macro . . . 230
JOBCOM (Job Communication) Macro . . . 231
LBRET (Label-Routine Return) Macro . . . 232
Checking Disk Extents . . . 232
Checking User Standard Labels on Disk . . . 232
Writing User Standard Labels on Disk . . . 232
Checking User Standard Tape Labels . . . 232
Writing User Standard Tape Labels . . . 233
Writing or Checking Nonstandard Tape Labels . . . 233
LBSERV (Control IBM 3494 Tape Library) Macro . . . 234
Overview of LBSERV Macro . . . 234
Reason Codes . . . 243
LFCB (Load Forms Control Buffer) Macro . . . 249
Return Codes in Register 15 . . . 251
LIBRDCB (Librarian Data Control Block) Macro . . . 252
Library Macro Notation . . . 253
LIBRM CLOSE (Close Library Member) Macro . . . 256
Return Codes . . . 257

LIBRM DELETE (Delete Library Member) Macro . . . 257
Return Codes . . . 258
LIBRM GET (Get Library Member) Macro . . . 259
Return Codes . . . 261
LIBRM LIBDEF (Define Sublibrary Chain) Macro . . . 262
Return Codes . . . 263
LIBRM LIBDROP (Drop Sublibrary Chain) Macro . . . 264
Return Codes . . . 264
LIBRM LOCK (Lock Library Member) Macro . . . 265
Return Codes . . . 266
LIBRM NOTE (Note Member Address) Macro . . . 266
Return Codes . . . 268
LIBRM OPEN (Open Library Member) Macro . . . 269
Return Codes . . . 272
LIBRM POINT (Point to Noted Member Record) Macro . . . 273
Return Codes . . . 274
LIBRM PUT (Put Library Member) Macro . . . 275
Return Codes . . . 276
LIBRM RENAME (Rename Library Member) Macro . . . 277
Return Codes . . . 278
LIBRM SHOWCB (Show Librarian Control Block) Macro . . . 279
LIBRM STATE CHAIN (Search Library Chain) Macro . . . 280
Return Codes . . . 281
LIBRM STATE LIB (Search Library) Macro . . . 283
Return Codes . . . 284
LIBRM STATE MEMBER (Search Library Member) Macro . . . 285
Return Codes . . . 288
LIBRM STATE SUBLIB (Search Sublibrary) Macro . . . 289
Return Codes . . . 291
LIBRM UNLOCK (Unlock Library Member) Macro . . . 292
Return Codes . . . 293
LITE (Pocket-Light Control) Macro . . . 294
LOAD (Load a Phase) Macro . . . 295
LOCK (Lock a Resource) Macro . . . 299
Return Codes in Register 15 . . . 301
MAPBDY (Map Boundary Information) Macro . . . 302
MAPBDYVR (Map Boundary Information) Macro . . . 303
MAPDNTRY (Map Directory Entry) Macro . . . 304
MAPEXTR (Map EXTRACT Service) Macro . . . 306
MAPSAVAR (Map Save Area) Macro . . . 309
MAPSSID (Map for SUBSID) Macro . . . 311
MAPSYSP (Map System Layout) Macro . . . 312
MAPXPCCB (Map Cross-Partition Control Block) Macro . . . 313
MODDTL (Modify DTL Block) Macro . . . 317
MRMOD (MICR Input Module Definition) Macro . . . 320
MVCOM (Move to Communication Region) Macro . . . 321
NOTE (Note-Address) Macro . . . 322
OPEN and OPENR (Open a File) Macro . . . 323
ORMOD (Optical Reader Input Module Definition) Macro . . . 324
Standard ORMOD Names . . . 325
Subset/Superset ORMOD Names . . . 325
PAGEIN (Page-In Request) Macro . . . 326
Return Information . . . 327
PDUMP (Partial-Dump Request) Macro . . . 328

PFIX (Page-Fix Request) Macro 329
Exceptional Conditions 330
Return Codes in Register 15 330
PFREE (Page-Free Request) Macro 331
Exceptional Conditions 332
Return Codes in Register 15 332
POINTR (Point to Noted Record) Macro 333
POINTS (Point to Start) Macro 333
POINTW (Point Behind Noted Record) Macro 334
POST (Post Event) Macro 335
PRMOD (Printer Output Module Definition) Macro 336
Standard PRMOD Names 338
Subset/Superset PRMOD Names 339
PRTOV (Printer Overflow Control) Macro 340
PUT (Put Record) Macro 341
PUTR (PUT with Reply) Macro 343
QSETPRT (Query Printer Setup) Macro 344
Return Codes 344
Calling SETPRT for a VSE/POWER-Controlled Printer 346
RCB (Resource Control Block Definition) Macro 348
RDLNE (Read a Line) Macro 348
READ (Read a Record) Macro 349
REALAD (Real Address Return) Macro 350
RELEASE (Release Logical Unit) Macro 351
RELPAG (Release Page) Macro 352
Exceptional Conditions 353
Return Codes in Register 15 353
RELSE (Release a Block) Macro 354
RESCN (Re-Scan) Macro 354
RETURN (Return after Call) Macro 356
RUNMODE (Run-Mode Indication) Macro 356
SAVE (Save Register) Macro 357
SDUMP/SDUMPX 357
Return Codes in Register 15 361
SECTVAL (Sector-Value Calculation) Macro 362
SEOV (System End-of-Volume) Macro 364
SETDEV (Set Device) Macro 364
SETFL (Set File Load Mode) Macro 365
SETIME (Set Interval Timer) Macro 365
SETL (Set Limits) Macro 367
SETPFA (Set Link to Page-Fault Appendage) Macro 369
General Coding Requirements 370
Register Usage 370
Entry Linkage 371
Page Fault Queue 371
Processing in the Appendage Routine 371
SETPRT (Set the Printer) Macro 373
SPLEVEL (Set and Test Macro Level) Macro 382
STXIT (Set Exit) Macro 383
SUBSID (Subsystem Information Display) Macro 389
Return Codes in Register 15 390
SYSSTATE (Set and Test Address Space Control Mode) Macro 391
TECB (Timer Event Control Block) Macro 392
TPIN (Telecommunication Priority In) Macro 393

TPOUT (Telecommunication Priority Out) Macro . . . 393
TRUNC (Truncate Block) Macro . . . 394
TTIMER (Test Interval Timer) Macro . . . 394
UNLOCK (Unlock Resource) Macro . . . 395
Return Codes in Register 15 . . . 395
VIRTAD (Virtual Address Return) Macro . . . 396
WAIT (Wait for Event) Macro . . . 397
WAITF (Wait for Completion of I/O) Macro . . . 399
WAITM (Wait for Multiple Events) Macro . . . 400
WRITE (Write a Record) Macro . . . 401
WTO (Write to Operator) Macro . . . 403
Return Codes in Register 15 . . . 408
Cancel Codes . . . 409
WTOR (Write to Operator with Reply) Macro . . . 410
Return Codes in Register 15 . . . 413
Cancel Codes . . . 413
XECBTAB (Cross-Partition Event Control Block Table) Macro . . . 414
Feedback Information . . . 416
XPCC (Cross-Partition Communication) Macro . . . 417
XPCCB (Cross-Partition Control Block) Macro . . . 421
XPOST (Cross-Partition Post) Macro . . . 423
Return Codes in Register 15 . . . 424
XWAIT (Cross-Partition Wait) Macro . . . 424
Return Codes in Register 15 . . . 425
YEAR224 Macro . . . 426

Appendix A. Control Character Codes . . . 429
CTLCHR=ASA Option . . . 429
CTLCHR=YES Option . . . 430
Stacker Selection Codes . . . 430
Printer Control Codes . . . 431

Appendix B. American National Standard Code for Information Interchange . . . 433

Appendix C. Standard and Non-Standard Labels . . . 439
Processing of User Labels . . . 439
Coding Requirements - User-Standard Labels . . . 440
Coding Requirements - Non-Standard Labels on Tape . . . 441
Formats of Volume and File Labels . . . 442
Volume Label on Disk (VOL1) . . . 442
IBM Standard File Labels on Disk . . . 442
User-Standard File Labels on Disk . . . 446
Volume Labels on Diskette . . . 448
IBM Standard File Labels on Diskette . . . 448
Volume Labels on Tape . . . 449
IBM Standard File Labels on Tape . . . 450
User-Standard File Labels on Tape . . . 454
Non-Standard File Labels on Tape . . . 454

Appendix D. Librarian Feedback Codes . . . 455

Appendix E. VSE Macros Intended for Customer Use . . . 459
VSE/Advanced Functions . . . 459

VSE/SP Unique Code 462
VSE/POWER 462
Execution Macros 462
Mapping Macros 462
VSE/Interactive Computing and Control Facility (VSE/ICCF) 462
VSE/Virtual Storage Access Method (VSE/VSAM) 464

Appendix F. VSE/ESA Macros And Their Mode Dependencies 465
VSE/ESA Downward-Compatible Macros 469

Glossary 471

Index 481

Figures

1. Subtask-Save Area (120 Bytes) . . . 25
2. Layout and Contents of the Command Control Block (CCB) . . . 57
3. MICR Document Buffer Format . . . 70
4. Character Set Option List . . . 86
5. Maximum and Initial BLOCKS Specification . . . 99
6. ASOCFLE Operand Usage with Print Associated Files . . . 109
7. Label Extent Information Field . . . 122
8. COREXIT Routine Functions . . . 127
9. DTFDU Error Options . . . 132
10. FilenameC-Status Byte if IOROUT Specifies ADD, RETRVE, or ADDRTR . . . 136
11. FilenameC-Status Byte if IOROUT=LOAD . . . 137
12. ERREXT Parameter List . . . 138
13. Output Area Requirements for Loading or Adding Records to a File by ISAM . . . 140
14. I/O Area Requirements for Random or Sequential Retrieval by ISAM . . . 141
15. Work Area Requirements . . . 144
16. Operands to Define a Checkpoint File on Disk . . . 160
17. Maximum and Assumed Lengths for the IOAREA1 in Number of Bytes 165
18. GETVCE Output Information . . . 215
19. Layout of the LBSERV-Generated DSECT . . . 223
20. Layout and Contents of the I/O Request Block (IORB) . . . 224
21. LBSERV : OPERANDS PER FUNCTION . . . 235
22. Volume Status in IJJLTSTA . . . 237
23. Media Type in IJJLTMED . . . 238
24. Device Status in IJJLTSTA . . . 239
25. Library Status in IJJLTSTA . . . 240
26. Reason Codes Generated by VGS . . . 243
27. Reason Codes Generated by VSE . . . 247
28. Operand Notation for LIBRM Requests . . . 254
29. Bit Configuration of the Pocket-Light Switch Area . . . 294
30. System Action for Control Definitions in DTLs . . . 300
31. System Actions by Return Code and FAIL Operand . . . 301
32. Layout of the MAPBDY-Generated DSECT . . . 302
33. Layout of the MAPBDYVR-Generated DSECT . . . 303
34. Layout of the MAPDNTRY-Generated DSECT for DE=VSE . . . 304
35. Layout of the MAPDNTRY-Generated DSECT for DE=YES . . . 305
36. Layout of the MAPEXTR-Generated DSECT for MODE=TEMP . . . 307
37. Layout of the MAPEXTR-Generated DSECT for MODE=PERM . . . 307
38. Layout of the MAPEXTR-Generated DSECT for MODE=SYSP . . . 308
39. Layout of the MAPEXTR-Generated DSECT for ID=ATLCUU . . . 308
40. Layout of the STXIT Save Area (AMODE=24 and MSGDATA=NO) . . . 309
41. Layout of the Extended STXIT Save Area (AMODE=ANY or MSGDATA=YES) . . . 310
42. Layout of MAPSSID-Generated DSECT . . . 311
43. Layout of the MAPSYSP-Generated DSECT . . . 312
44. MAPXPCCB Macro Return Codes (IJBXRETC) . . . 314
45. MAPXPCCB Reason Codes (IJBXREAS) . . . 316
46. MAPXPCCB Function Codes (IJBXFCT) . . . 316
47. MAPXPCCB Function Descriptor Codes (IJBXFDSC) . . . 317

© Copyright IBM Corp. 1990, 1999

48. SDUMP Reason Codes for Return Code 8 361
49. Field Supplied for SETL Processing by Record ID 368
50. Internal Page-Fault Queue and Communication with the System 372
51. Effect of an AB, IT, OC, or PC Interrupt During STXIT Routine Execution 388
52. XECBTAB Feedback Information 416
53. Coding Example Showing the Use of XECBTAB with TYPE=CHECK and XWAIT 426
54. Syntax of YEAR224 Macro 426
55. ASCII Character Set 434
56. ASCII to EBCDIC Correspondence 435
57. Disk Volume Label (VOL1) 442
58. IBM Standard Disk File Label (Format-1) 443
59. IBM Standard Disk File Continuation Label (Format-3) 445
60. Disk VTOC Label (Format-4) 445
61. User-Standard Disk-File Label (Header and Trailer) 447
62. User-Standard Disk-File Label (Five UHLs and Four UTLs are Specified) 447
63. User-Standard Disk-File Label (Three UHLs are Specified) 448
64. Diskette Volume Label 448
65. Diskette File Label 448
66. Tape Volume Label for EBCDIC Code 449
67. Tape Volume Label for ASCII Code 450
68. First IBM Standard Tape File Label for EBCDIC Code 451
69. First IBM Standard Tape File Label for ASCII Code 451
70. Second IBM Standard Tape File Label for ASCII Code 451
71. Second IBM Standard Tape File Label for EBCDIC Code 452
72. Librarian Feedback Codes 456
73. VSE/ESA Macros and Their Mode Dependencies (Execution Time) . . 466

Notices

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

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, North Castle Drive, Armonk, NY 10504-1785, U.S.A.

Any pointers in this publication to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement. IBM accepts no responsibility for the content or use of non-IBM Web sites specifically mentioned in this publication or accessed through an IBM Web site that is mentioned in this publication.

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:

```
IBM Deutschland Informationssysteme GmbH
Department 0215
Pascalstr. 100
70569 Stuttgart
Germany
```

Such information may be available, subject to appropriate terms and conditions, including in some cases payment of a fee.

Programming Interface Information

This manual is intended as a reference source for programmers using the macro support available with IBM VSE/Enterprise Systems Architecture (VSE/ESA). It contains a complete description of all VSE/ESA data management (IOCS) and system control macros.

This manual documents intended Programming Interfaces that allow the customer to write programs to obtain the services of VSE/ESA.

© Copyright IBM Corp. 1990, 1999

Trademarks and Service Marks

The following terms are trademarks of International Business Machines Corporation in the United States, or other countries, or both:

ACF/VTAM
CICS
CICS/VSE
DFSMS/VM
ECKD
Enterprise Systems Architecture/370
Enterprise Systems Architecture/390
ESA/370
ESA/390
MVS/ESA
System/370
Virtual Machine/Enterprise Systems Architecture
VM/ESA
VSE/ESA
VTAM

About This Book

This manual is intended as a reference source for programmers using the macro support available with VSE/Enterprise Systems Architecture (VSE/ESA). These macros can be used in application programs (or routines of such programs) written in assembler language.

The manual documents reference information about the two types of macros that VSE/ESA offers: data-management or input/output (IOCS) macros and control program macros. The publication lists the macros in alphabetic order of their names.

Who Should Use This Book

This manual is mainly intended for programmers writing application programs in assembler language.

Where to Find More Information

For the most part, programming details have been omitted in order to provide rapid access to the information in this publication. If the publication does not meet your information needs, refer to the IBM publications:

VSE/ESA System Macro User's Guide
VSE/ESA Extended Addressability

To define a sequential file in VSAM-managed space using the file definition macro DTFSD, you should consult also the IBM publication:

VSE/VSAM User's Guide and Application Programming

To assemble and link-edit your program (or routine), you may have to consult the IBM publications:

VSE/ESA Guide to System Functions
VSE/ESA System Control Statements

Programmed tools available with VSE/ESA to help you debug your program are described in the IBM publication:

VSE/ESA Diagnosis Tools

For planning and migration information, refer to the chapter “Planning for Migration” in the *VSE/ESA Planning* manual.

The High Level Assembler is described in the following manuals:

Programmer's Guide
Language Reference

© Copyright IBM Corp. 1990, 1999

VSE/ESA Home Page

VSE/ESA has a home page on the World Wide Web, which offers up-to-date information about VSE-related products and services, new VSE/ESA functions, and other items of interest to VSE users.

You can find the VSE/ESA home page at:

`http://www.ibm.com/s390/vse/`

Summary of Changes

This manual has been updated to reflect changes and enhancements for VSE/ESA 2.2, VSE/ESA 2.3, and VSE/ESA 2.4. It also includes terminology, maintenance, and editorial changes.

© Copyright IBM Corp. 1990, 1999

Chapter 1. Using the Macros

To request system services, programs use macros. All macros described in this manual are written in assembler format statements; they consist of a number of fields as discussed under Macro Notation. When you code a macro, the assembler processes it by using the macro definitions supplied by IBM and stored in a sublibrary when the system is generated.

The assembler expands the macro into executable machine instructions and/or data fields in the form of assembler language statements. The executable machine instructions typically consist of a branch around the data fields, instructions that load registers, and an instruction that gives control to the system. The macro expansion appears as part of the assembler output listing.

The data fields which are derived from the macro operands are used at execution time by the control program routine that performs the VSE service associated with the macro.

VSE/ESA offers two different types of macros: data management (IOCS) and system control macros. The data management macros define the characteristics of a file and identify the I/O operation to be performed on the file. The system control macros enable you to make use of control functions available under VSE/ESA.

Selecting the Macro Level

VSE/ESA Version 1.3/2.1 supports all VSE/ESA Version 1.1/1.2 macros. Therefore, programs that issue macros and that run on a Version 1.1/1.2 system should also run on a Version 1.3/2.1 system (provided AMODE=24 and RMODE=24).

Starting with VSE/ESA 1.3, 31-bit addressing support was introduced.

There are, however, Version 1.3/2.1 macros that cannot execute on Version 1.1/1.2. This means that programs running on VSE/ESA 1.3/2.1 and issuing these macros might not run on VSE/ESA 1.1/1.2, because a Version 1.1/1.2 system cannot process all the macro parameters that work on a Version 1.3/2.1 system. When you try to run a Version 1.3/2.1 program on a Version 1.1/1.2 system, the program might not execute as expected. The macros in question are called *downward incompatible*. In general, macros with new parameters are downward incompatible, if not stated otherwise in the description. (For a list of the downward **compatible** macros, see “VSE/ESA Downward-Compatible Macros” on page 469).

For the following macros it is possible to generate downward-compatible macro *expansions* by using the SPLEVEL macro:

```
FCEPGOUT
PAGEIN
PFIX
PFREE
RELPAG
WTO
WTOR
```

© Copyright IBM Corp. 1990, 1999

The SPLEVEL macro sets (or tests) a global symbol that is interrogated by these macros during assembly to determine the type of expansion to be generated. For details refer to the description of the SPLEVEL macro on page 382

Addressing Mode and the Macros

A program can execute either in 24-bit addressing mode (AMODE 24) or in 31-bit addressing mode (AMODE 31). Among the macros described in this manual, there is one group that has no requirements on the addressing mode in which a program executes. There is, however, another group that requires the program to be executing in 24-bit addressing mode and the parameters to be passed in 24-bit addressable storage (that is, below 16MB). This is indicated individually for each macro under *Requirements for the caller* immediately after the macro's syntax description. It is also indicated in Appendix F of this manual, which lists all macros together with their mode dependencies.

In general, a program executing in 24-bit addressing mode cannot pass parameter addresses that are higher than 16MB. However, there are exceptions; for example, a program executing in 24-bit addressing mode can:

- Free storage above 16MB using the FREEVIS macro
- Allocate storage above 16MB using the GETVIS macro.

If a program running in 31-bit addressing mode issues a macro whose RMODE (residency mode) is ANY, parameter addresses can be above or below the 16MB line. Macros with RMODE 24, on the other hand, require parameter addresses below 16MB. The required RMODE of each macro is also indicated in the individual macro description (and in Appendix F).

A program running in 31-bit addressing mode must be recompiled on Version 1.3/2.1 libraries and use (via SPLEVEL) the VSE/ESA Version 1.3/2.1 macro expansion of the following macros:

```
FCEPGOUT
PAGEIN
PFIX
PFREE
RELPAG
```

For details on 31-bit addressing, AMODE, RMODE, and other related subjects, see the manual *VSE/ESA Extended Addressability* under “Introducing AMODE and RMODE”.

Address Space Control (ASC) Mode

A program can execute in either **primary** or **AR** (access register) ASC mode. For details, see the manual *VSE/ESA Extended Addressability* under “Basic Concepts”.

Some VSE/ESA macros (DSPSERV, SDUMP, SDUMPX, for example) can generate code that is appropriate for programs in either primary or AR mode. This is also indicated in the individual macro description (and in Appendix F). A global variable tells these macros which type of code to generate. The SYSSTATE macro allows you to test or to set this variable.

When you assemble a program, the initial value of this variable indicates primary ASC mode. If you do not change the variable, macros that test it will generate code appropriate for primary ASC mode. Thus, if your program receives control in primary ASC mode, you do not need to change the variable. If, however, your program receives control in AR ASC mode, you might have to issue SYSSTATE ASCENV=AR before issuing any macro that tests the variable. To ensure that your programs always generate code appropriate for their ASC mode, IBM recommends that:

- **All** programs that use macros should issue SYSSTATE before issuing any other macros. Programs in primary ASC mode must issue SYSSTATE ASCENV=P. Programs in AR mode must issue SYSSTATE ASCENV=AR.
- If your program switches from one ASC mode to another, issue SYSSTATE immediately after the mode switch to indicate the new ASC mode.

Once a program has issued SYSSTATE, there is no need to reissue it unless the program switches ASC mode.

Using X-Macros

Some VSE macros (at present SDUMPX only) support callers in both primary and AR ASC mode. When the caller is in AR mode, the macro must generate larger parameter lists. Some services (at present only the dump service) offer two macros: one for callers in primary mode and one for callers in AR mode. The name of the macro for the AR mode caller is the same as the name of the macro for primary mode callers, except that the AR mode macro name ends with an "X" (SDUMPX vs. SDUMP).

The only way an X-macro knows that a caller is in AR mode is by checking the global symbol that the SYSSTATE macro sets. If SYSSTATE ASCENV=AR has been issued, the macro generates code that is valid for callers in AR mode. If it has not been issued, the macro generates code that is not valid for callers in AR mode. When your program returns to primary mode, use the SYSSTATE ASCENV=P macro to reset the global symbol.

The rules for an X-macro are:

- Callers in primary mode can invoke either macro (X or non-X).

  Some parameters on the X-macros, however, are not valid for callers in primary mode. Some parameters on the non-X macros are not valid for callers in AR mode. Check the macro descriptions in this manual for these exceptions.

- Callers in AR mode should issue the X-macros.

  If a caller in AR mode issues the non-X macro, the system substitutes the X-macro and sends a message describing the substitution.

Passing Parameters in AR Mode

All macros which can be issued in AR mode and which include control parameters, place these parameters in the primary address space.

Register Usage

Registers 2 through 12 are available for general use. However, the PUTR (PUT with Reply) macro makes use of register 2. General registers 0, 1, 13, 14, and 15 are available to your program only under certain conditions.

The following paragraphs describe the general uses of these registers by IOCS, but the description is not meant to be all inclusive. Certain applications, such as a MICR stacker selection routine, may require different registers.

- Registers 0, 1, and 15

  IBM supplied macros use these registers to pass parameters and return codes. Therefore, the registers may be used without restriction only for immediate computations.

- Register 13

  System routines, and also IOCS routines, use this register as a pointer to a 72-byte save area. When using the CALL, SAVE, or RETURN macro, you can set the address of the save area at the beginning of each phase of your program, and leave it unchanged thereafter. However, if reentrant, read-only code is shared among tasks, register 13 must contain the address of another save area to be used by that code each time the code is used by another task.

- Registers 14 and 15

  IOCS uses these registers for linkage without saving their contents. If you use the registers, either save their contents (and reload them later) or finish with these registers before IOCS uses them.

  Not all logic modules use standard save area conventions. Therefore, if you use a read-only logic module (supplying a module save area) in a subroutine, the save area back-chain pointer can get lost.

- Floating-Point Registers

  If your program uses floating-point registers in a subroutine, ensure that this subroutine:

  1. Saves their contents when it receives control.
  2. Restores their contents when it returns control.

Macro Notation

Macros, like assembler statements, have a name field, operation field and operand field. Comments can also be included as in assembler statements, although certain macros require a comment to be preceded by a comma if the macro is issued without an operand. These macros are: CANCEL, DETACH, FREEVIS, GETIME, GETVIS, and TTIMER.

The **name field** in a macro may contain a symbolic name. Some macros (for example, CCB, TECB, or DTFxx) require a name.

The **operation field** must contain the mnemonic operation code of the macro.

The operands in the **operand field** must be written in either positional, keyword, or mixed format.

There must be no comma between the operation and the operand field; that is, the first operand must not start with a comma.

Positional Operands

In this format, the operand values must be in the exact order shown in this publication. Each operand, except the last, must be followed by a comma; no embedded blanks are allowed. If an operand is to be omitted in the macro, and following operands are included, a comma must be inserted to indicate the omission. No commas need to be included after the last operand. Column 72 must contain a continuation punch (any non-blank character) if the operands fill the operand field and overflow onto another line.

The macro GET, for example, uses the positional format. A GET for a file named CDFILE using a work area named WORK is written as follows:

```
GET CDFILE,WORK
```

Keyword Operands

An operand written in keyword format can have this form:

```
LABADDR=MYLABELS
```

where:

LABADDR is the keyword
MYLABELS is a name you specify
LABADDR=MYLABELS is the complete operand.

The keyword operands in the macro may appear in any order, and those that are not required may be omitted. Different keyword operands may be written in the same statement, each followed by a comma, except for the last operand of the macro.

Mixed Format

The operand list contains both positional and keyword operands. The keyword operands can be written in any order, but they must be written to the right of any positional operands in the macro.

For more detailed information on coding macro statements, see the *Assembler Language* manual.

Comments in Macros

You can include a comment in a macro in the same way as in an assembler language instruction. However, a comment together with a macro that has no operand requires that your comment begins with a comma.

Understanding Syntax Diagrams

This section describes how to read the syntax diagrams in this manual.

To read a syntax diagram follow the path of the line. Read from left to right and top to bottom.

- The ►►— symbol indicates the beginning of a syntax diagram.

- The ⟶ symbol, at the end of a line, indicates that the syntax diagram continues on the next line.
- The ⟶ symbol, at the beginning of a line, indicates that a syntax diagram continues from the previous line.
- The ⟶◂ symbol indicates the end of a syntax diagram.

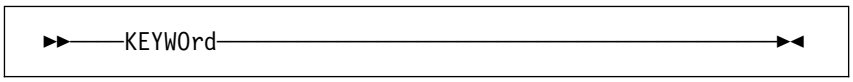
Syntax items (for example, a keyword or variable) may be:

- Directly on the line (required)
- Above the line (default)
- Below the line (optional)

Uppercase Letters

Uppercase letters denote the shortest possible abbreviation. If an item appears entirely in uppercase letters, it can not be abbreviated.

You can type the item in uppercase letters, lowercase letters, or any combination. For example:



In this example, you can enter KEYWO, KEYWOR, or KEYWORD in any combination of uppercase and lowercase letters.

Symbols

You **must** code these symbols exactly as they appear in the syntax diagram

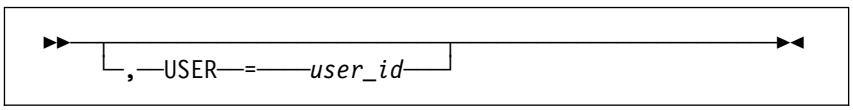
*	Asterisk
:	Colon
,	Comma
=	Equal Sign
-	Hyphen
//	Double slash
()	Parenthesis
.	Period
+	Add

For example:

```
* $$ LST
```

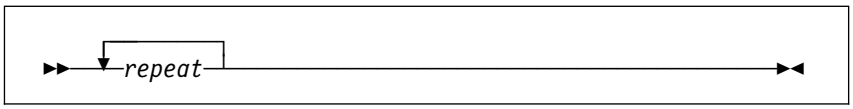
Variables

Lowercase letters written in *italics* denote variable information that you must substitute with specific information. For example:

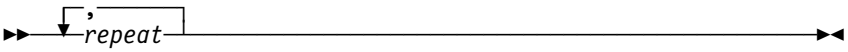


Here you must code `USER=` as shown and supply an ID for user_id. You may, of course, enter `USER` in lowercase, but you must not change it otherwise.

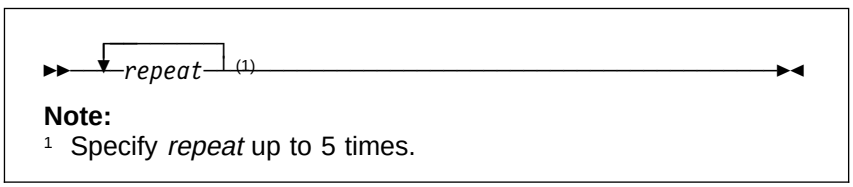
Repetition An arrow returning to the left means that the item can be repeated.



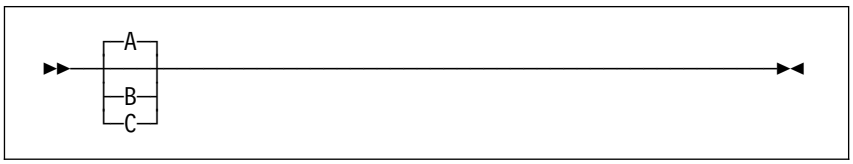
A character within the arrow means you must separate repeated items with that character.



A footnote (1) by the arrow references a limit that tells how many times the item can be repeated.



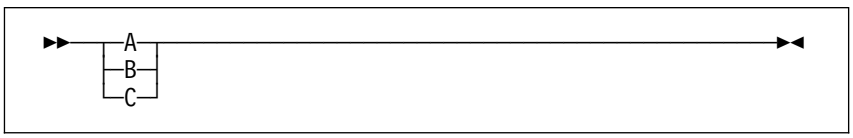
Defaults Defaults are above the line. The system uses the default unless you override it. You can override the default by coding an option from the stack below the line. For example:



In this example, A is the default. You can override `A` by choosing `B` or `C`.

Required Choices

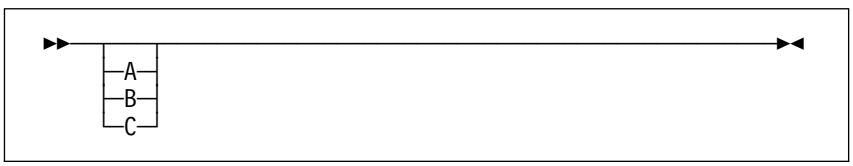
When two or more items are in a stack and one of them is on the line, you **must** specify one item. For example:



Here you must enter either `A` or `B` or `C`.

Optional Choice

When an item is below the line, the item is optional. Only one item **may** be chosen. For example:



Here you may enter either `A` or `B` or `C`, or you may omit the field.

Required Blank Space

A required blank space is indicated as such in the notation. For example:

```
 * $$ EOJ
```

This indicates that at least one blank is required before and after the characters $$.

Register Notation

Certain operands can be specified in either of two ways:

- You may specify the operand directly – as a symbol, for instance. This results in code that, for example, cannot be executed in the SVA because it is not reentrant.
- You may load the address of the value into a register before issuing the macro. This way, which is called register notation, results in reentrant code that may be executed in the SVA. When using register notation, the register should contain only the specific address; high-order bits should be set to 0.

A typical example of an operand that allows register notation is the specification of a file name in the GET or PUT macro. The operand is represented in this manual as follows:

```
filename|(rn)
```

n = A decimal number indicating the sequence of specifications using register notation.

When the macro is assembled, instructions are generated to pass the information contained in the specified register to IOCS or to the supervisor. For example, if an operand is written as (8), IOCS or the supervisor expects information to be stored at the address contained in general register 8. This is an example of ordinary register notation.

You can save both storage and execution time by using what is known as special register notation. In this method, the operand is shown in the format description of the macro as either (0) or (1), for example. This notation is special because the use of registers 0 and 1 is allowed only for the indicated purpose.

If special register notation is indicated by (0) or (1) in a macro format description and you use ordinary register notation, the macro expansion will contain an extra LR instruction, for example, LR 0,8.

The format description for each macro shows whether special register notation can be used and for which operands. The following example indicates that the filename operand can be written as (1) and the workname operand as (0):

```
GET filename|(1),workname|,(0)
```

If either of these special register notations is used, your program must load the designated register before executing the macro expansion. Ordinary register notation can also be used.

Operand Notation

Certain system control macros (for instance, ATTACH, GENIORB, GENL, LOAD) allow three notations for an operand:

- Register notation

  This is described in the preceding paragraph.

- Notation as a relocatable expression

  In the macro expansion, this results in an A-type address constant.

- Notation in the form (S,address)

  In the macro expansion, this results in the generation of an address in base-displacement form. You can specify the address in either of the following ways:

  – As a relocatable expression; for example: (S,RELOC).
  – As two absolute expressions, the first of which represents the displacement and the second the base register; for example: (S,512(12)).

  Consider using this notation if your program is to be reenterable. In a reenterable program, macro operands often refer to fields in dynamic storage. The (S,address) format offers an alternative to register notation: if two or more of such operands have to be provided for one macro, there is no need for loading addresses into that many registers.

Chapter 2. Macro Descriptions

This section describes the macros in alphabetical order of their names. For each macro, the section gives the format of the macro and a summary of the macro's function, followed by a description of the macro's operand(s).

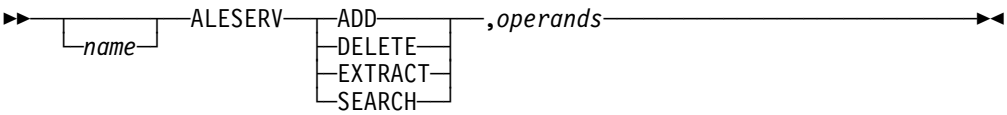
ALESERV (Access List Entry) Macro

The ALESERV macro manages the contents of access lists. An access list is a table in which each entry identifies a data space to which one or more programs have access. Each entry in the table is referenced by an ALET (access list entry token).

For detailed guide information on how to create and use data spaces, see “Chapter 8, Creating and Using Data Spaces” in the manual *VSE/ESA Extended Addressability*.

For definitions of the terms used with the ALESERV macro, see the Glossary at the back of this manual.

The ALESERV macro supports the following main functions:



ADD Add an entry to an access list

DELETE
Delete an entry from an access list

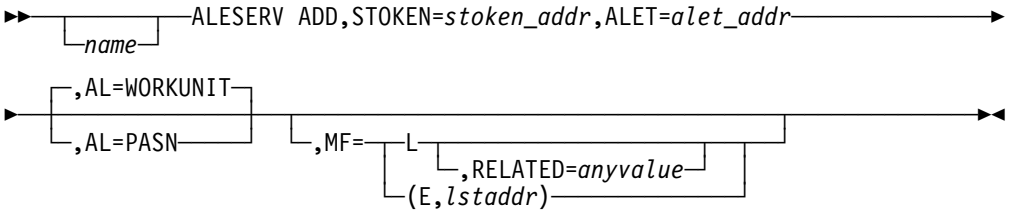
EXTRACT
Obtain the STOKEN for a specified ALET

SEARCH
Locate an ALET for a specified STOKEN

For a detailed description of the main functions, see the “ALESERV (Access List Entry) Macro” with the corresponding keyword (ALESERV ADD, ALESERV DELETE,...).

© Copyright IBM Corp. 1990, 1999

ALESERV ADD (Add Access List Entry) Macro



Requirements for the caller:

AMODE: 24 or 31

RMODE: 24 or ANY

ASC Mode: Primary or AR (access register)

The ALESERV ADD macro adds an entry to an access list and returns the ALET that references this entry.

If you want to know whether a data space already has an entry on an access list, use the ALESERV SEARCH macro.

STOKEN=stoken_addr
Specifies the address of the 8-byte identifier of the data space that the program wants to access and for which the entry is to be added. You might have received the STOKEN as output from the DSPSERV CREATE macro or from another user.

ALET=alet_addr
Specifies the address of the location where the system returns the 4-byte ALET for the access list entry that the system added.

AL=WORKUNIT | PASN
WORKUNIT specifies that the access list to which the entry is to be added is a 'dispatchable unit access list' (DU-AL), that is, an access list associated with a VSE task. PASN specifies that the access list is a 'primary address space access list' (PASN-AL), that is, an access list associated with a partition.

Use AL=WORKUNIT if you want to limit the sharing of the data space to programs running under the owning task.

Use AL=PASN if you want other programs running in the partition to have access to the data space, or if you are adding an entry for a data space that has been created with DSPSERV SCOPE=COMMON.

MF=L...
L specifies the **list** form of the macro, which is used to construct a non-executable control program parameter list.

RELATED=anyvalue specifies any valid macro parameter expression which can be freely chosen by the user.

No other parameters may be specified if the list form of the macro is chosen.

MF=E...

E specifies the **execute** form of the macro, which uses the parameter list generated by the list form of the macro.

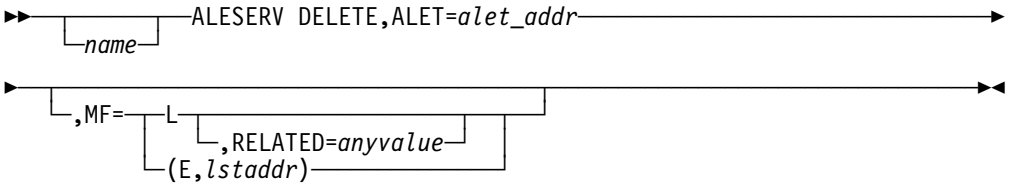
lstaddr specifies the address of the parameter list. This address must not be in a data space. If the caller of the macro is in 24-bit addressing mode, the address of the parameter list must not be above the 16MB line.

If the MF operand is omitted, the **standard** form of the macro is used, which places the parameters into an inline parameter list.

Return Codes in Register 15

00 Successful completion.
0C The current access list cannot be expanded. There are no free ALEs and the maximum size has been reached.
10 ALESERV could not obtain storage for an expanded access list.
18 The caller tried to add to the PASN-AL without being in PSW key-0 state.
38 The input STOKEN is invalid.
4C The space represented by the input STOKEN is invalid for cross-memory access.
5C The caller is not authorized to add a data space to an access list.
6C The caller tried to add an entry for a SCOPE=COMMON data space to a DU-AL.

ALESERV DELETE (Delete Access List Entry) Macro



Requirements for the caller:

AMODE: 24 or 31

RMODE: 24 or ANY

ASC Mode: Primary or AR (access register)

The ALESERV DELETE macro deletes an entry from an access list. After the access list entry has been removed, the connection between the ALET and the data space no longer exists.

Since the system does not check and notify programs about the reuse of an ALET, the program deleting an access list entry must ensure that other programs do not use the old ALET.

ALET=alet_addr
Specifies the address of the ALET for the access list entry to be deleted.

MF=L...
L specifies the **list** form of the macro, which is used to construct a non-executable control program parameter list.

RELATED=anyvalue specifies any valid macro parameter expression which can be freely chosen by the user.

No other parameters may be specified if the list form of the macro is chosen.

MF=E...
E specifies the **execute** form of the macro, which uses the parameter list generated by the list form of the macro.

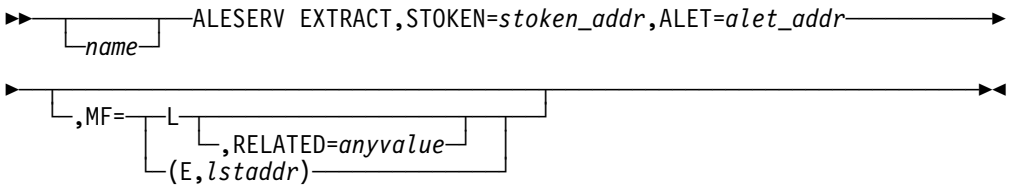
lstaddr specifies the address of the parameter list. This address must not be in a data space. If the caller of the macro is in 24-bit addressing mode, the address of the parameter list must not be above the 16MB line.

If the MF operand is omitted, the **standard** form of the macro is used, which places the parameters into an inline parameter list.

Return Codes in Register 15

00 Successful completion.
14 The input ALET corresponds to an invalid access list entry.
28 The caller specified an invalid ALET.
2C The caller attempted to delete an ALET reserved for system use.
30 The caller tried to delete an entry from the PSN-AL without being in PSW key-0 state.

ALESERV EXTRACT (Find a STOKEN) Macro



Requirements for the caller:

AMODE: 24 or 31

RMODE: 24 or ANY

ASC Mode: Primary or AR (access register)

The ALESERV EXTRACT macro requests that the system finds the STOKEN associated with the specified ALET. The caller can obtain the STOKEN for any space that is represented by a valid entry on the current access list (DU-AL or PASN-AL related to the current task/partition).

STOKEN=stoken_addr
Specifies the address of the location where the system is to return the 8-byte STOKEN that corresponds to the specified ALET.

ALET=alet_addr
Specifies the address of the location where the 4-byte ALET is given.

MF=L...
L specifies the **list** form of the macro, which is used to construct a non-executable control program parameter list.

RELATED=anyvalue specifies any valid macro parameter expression which can be freely chosen by the user.

No other parameters may be specified if the list form of the macro is chosen.

MF=E...
E specifies the **execute** form of the macro, which uses the parameter list generated by the list form of the macro.

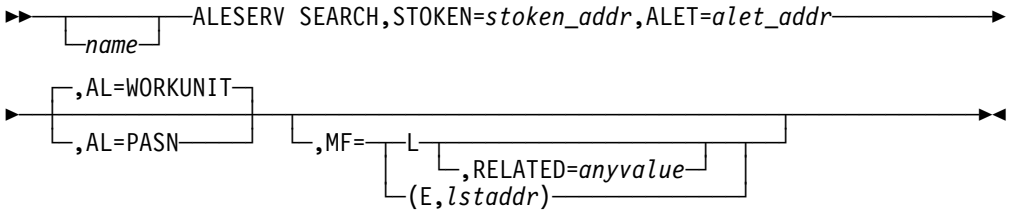
lstaddr specifies the address of the parameter list. This address must not be in a data space. If the caller of the macro is in 24-bit addressing mode, the address of the parameter list must not be above the 16MB line.

If the MF operand is omitted, the **standard** form of the macro is used, which places the parameters into an inline parameter list.

Return Codes in Register 15

00 Successful completion.
14 The input ALET corresponds to an invalid access list entry.
28 The caller specified an invalid ALET.
3C An ALET value of 1 was specified.
44 The ALE associated with the input ALET represents addressing capability to a deleted or terminated space.
58 The access list associated with the input ALET does not exist.

ALESERV SEARCH (Search Access List Entry) Macro



Requirements for the caller:

AMODE: 24 or 31

RMODE: 24 or ANY

ASC Mode: Primary or AR (access register)

The ALESERV SEARCH macro searches through the DU-AL or PASN-AL related to the current task/partition for an ALET that corresponds to the specified STOKEN. If the entry is on the list, the system returns the ALET. Otherwise, a return code is set in register 15.

STOKEN=stoken_addr
Specifies the address of the 8-byte STOKEN for which the system is to return the corresponding ALET.

ALET=alet_addr
As input to the SEARCH request, ALET specifies the point in the access list where the system is to begin the search. The following values are valid as start addresses:

- Minus one (-1) - Start at the beginning of the DU-AL or PASN-AL.
- Valid ALET - Start the search with the next ALET in the access list. It is recommended to start searching from the beginning of the access list, that is with ALET=-1. Starting with a valid ALET gives consistent results only if the program can ensure that no ADD or DELETE requests are executed while processing the SEARCH request.

As output from the SEARCH request, ALET specifies the address of the location where the system is to return the 4-byte ALET, if present. Otherwise, ALET is unchanged and register 15 contains a return code indicating that an ALET for the specified STOKEN is not on the access list.

AL=WORKUNIT | PASN
WORKUNIT specifies that the access list to be searched is a 'dispatchable unit access list' (DU-AL), that is, an access list associated with a VSE task. PASN specifies that the access list to be searched is a 'primary address space access list' (PASN-AL), that is, an access list associated with a VSE partition.

MF=L...
L specifies the **list** form of the macro, which is used to construct a non-executable control program parameter list.

RELATED=anyvalue specifies any valid macro parameter expression which can be freely chosen by the user.

No other parameters may be specified if the list form of the macro is chosen.

MF=E...

E specifies the **execute** form of the macro, which uses the parameter list generated by the list form of the macro.

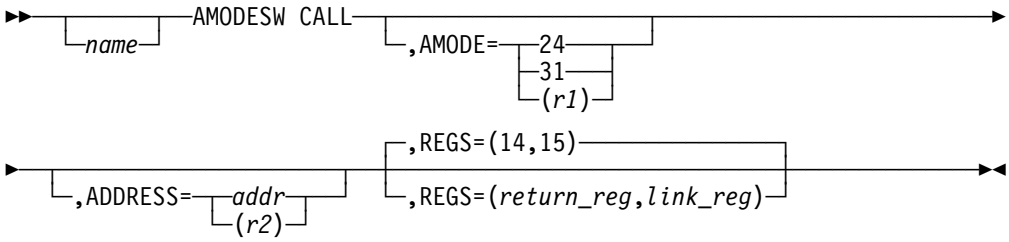
lstaddr specifies the address of the parameter list. This address must not be in a data space. If the caller of the macro is in 24-bit addressing mode, the address of the parameter list must not be above the 16MB line.

If the MF operand is omitted, the **standard** form of the macro is used, which places the parameters into an inline parameter list.

Return Codes in Register 15

00 Successful completion.
28 The caller specified an ALET that is not valid on the specified access list.
34 The caller specified an STOKEN that is not represented on the specified access list.
48 The caller specified AL=WORKUNIT, but the input ALET indexes into the PASN-AL or the caller specified AL=PASN and the input ALET indexes into the DU-AL.

AMODESW CALL (Addressing Mode Switch) Macro



Requirements for the caller:

AMODE: 24 or 31

RMODE: 24 or ANY

ASC Mode: Primary

The macro calls a subroutine and switches the addressing mode.

CALL Indicates that a subroutine is to be called and that the addressing mode is to be switched.

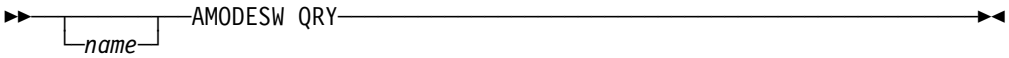
AMODE=24 | 31 | (r1)
Calls the subroutine and switches either to 24-bit or 31-bit addressing mode or sets the addressing mode according to the value of bit 0 of the specified register. The register must not be the same as the one used with the ADDRESS operand or the register used as the return register. If you do not specify AMODE, VSE sets the addressing mode as follows:

- If you specify ADDRESS=(reg), VSE obtains the new addressing mode from bit 0 of (reg). If you specify ADDRESS=addr, VSE obtains the new addressing mode from attributes declared with the AMODE assembler pseudo-op.
- If you do not specify the AMODE or the ADDRESS operand, VSE obtains the new addressing mode from bit 0 of the linkage register (specified in the REGS operand).

ADDRESS=addr | (r2)
Specifies the address, either directly or in a register (1-15), where control is to be transferred. If you omit the ADDRESS operand, VSE passes control to the address in the linkage register (specified in the REGS operand).

REGS=(return_reg|14,link_reg|15)
Specifies the linkage registers for this call. Valid registers are 1-15. If you do not specify REGS, VSE uses register 14 as **return_reg** and register 15 as **link_reg**: REGS=(14,15).

AMODESW QRY (Query Addressing Mode) Macro



Requirements for the caller:

AMODE: 24 or 31

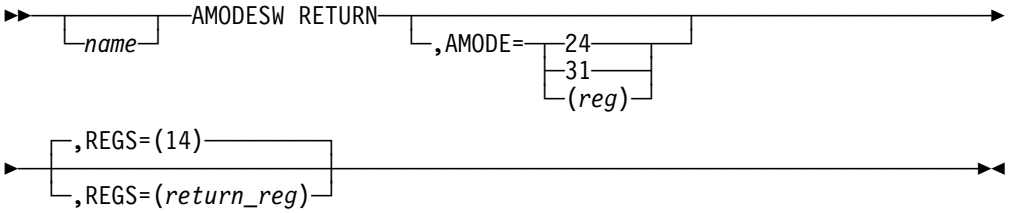
RMODE: 24 or ANY

ASC Mode: Primary

The macro can be used to determine the current addressing mode of a program.

QRY Determines the task's current addressing mode. Upon completion, register 1 contains either all 0s for 24-bit addressing mode or non-zero (X'80000000') for 31-bit addressing mode. (Register 1 is the only register that is being altered.)

AMODESW RETURN (Return from Subroutine) Macro



Requirements for the caller:

AMODE: 24 or 31

RMODE: 24 or ANY

ASC Mode: Primary

The macro makes a return to the caller of a subroutine.

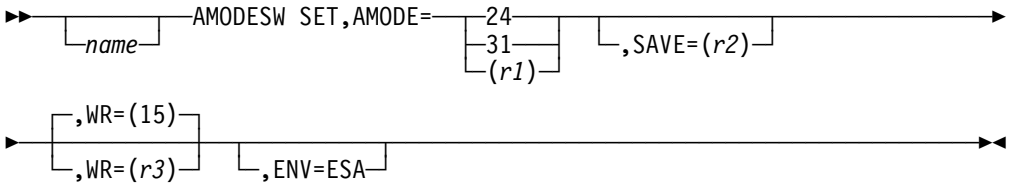
RETURN
Indicates that a subroutine is to return to its caller.

AMODE=24 | 31 | (reg)
Specifies that the subroutine is to return either in 24-bit or 31-bit addressing mode or in the addressing mode corresponding to the value of bit 0 of the specified register. The register must not be the same as the one used as return register (14 or the register specified in the REG operand).

If you do not specify AMODE, VSE sets the addressing mode according to the value of bit 0 of the return register specified in the REG operand.

REG=(return_reg|14)
Specifies the register that contains the address (and, optionally, the addressing mode) where control is to be returned. If you do not specify REG, VSE uses register 14 as the return register.

AMODESW SET (Set Addressing Mode) Macro



Requirements for the caller:

AMODE: 24 or 31

RMODE: 24 or ANY

ASC Mode: Primary

The macro can be used to change a program's addressing mode without branching to a subroutine. If used as a supervisor generation macro, the macro can be used to switch dynamic address translation either on or off.

SET Indicates that the program's addressing mode is to be changed.

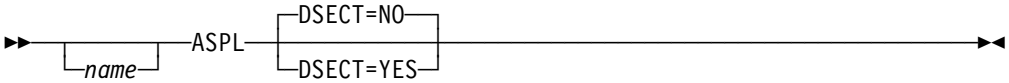
AMODE=24 | 31 | (r1)
Switches either to 24-bit or 31-bit addressing mode or sets the addressing mode according to the value of bit 0 of the specified register.

SAVE=(r2)
Saves the current (unknown) addressing mode in bit 0 of the specified register (1-14). If you do not specify SAVE, the current mode is not saved.

WR=(r3|15)
Specifies a work register. The contents of this register will be changed. If the operand is omitted, register 15 is taken as default.

ENV=ESA
Causes the system not to check the current environment (and assume ESA).

ASPL (Assign Parameter List) Macro

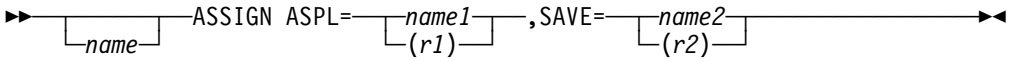


Required RMODE: **24**

The macro generates a 7-byte parameter list that is used to pass information to the ASSIGN macro. For the format of the parameter list, see “Assigning and Releasing an I/O Unit” in the *VSE/ESA System Macro User's Guide*.

DSECT=NO | YES
: Specify DSECT=YES if you want the parameter list to be generated as a mapping DSECT. If the operand is omitted, inline code is generated.

ASSIGN (Assign I/O Device) Macro



Requirements for the caller:

AMODE: 24

RMODE: 24

ASC Mode: Primary

The macro is used to dynamically assign and unassign tape, disk, and unit-record devices. The system will select a free or specified unit and assign it to a free or specified programmer logical unit. (Not all functions are defined for all device types; for details, see the ASPL parameter list under “Assigning and Releasing an I/O Unit” in the *VSE/ESA System Macro User's Guide*.)

When it has made the assignment, the system returns to your program the logical and physical unit numbers of the assigned unit. This information can be used by the RELEASE macro to release a unit dynamically when it is no longer needed.

A skeleton example that shows how tape drives are assigned and unassigned dynamically is also given under “Assigning and Releasing an I/O Unit” in the *VSE/ESA System Macro User's Guide*.

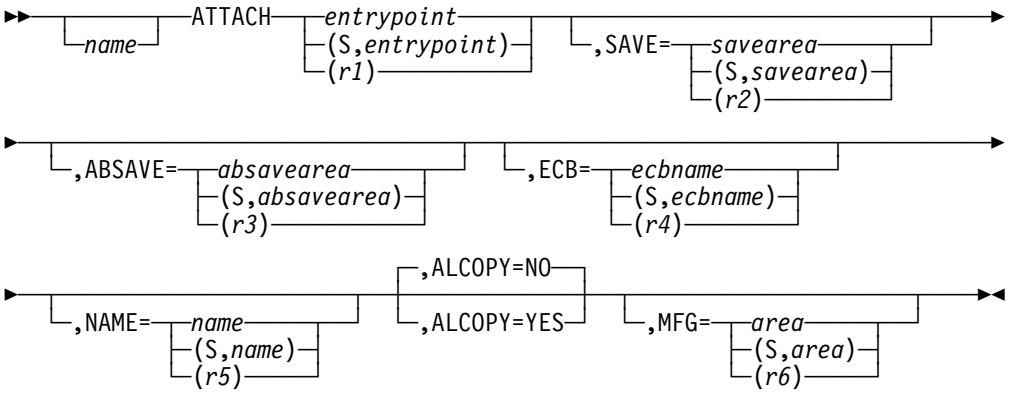
ASPL=name1 | (r1)
Specifies the address of the parameter list, in which you indicate the function (assign or unassign) to be performed. Use the mapping DSECT generated by the ASPL macro to interpret the fields in the parameter list.

SAVE=name2 | (r2)
Specifies a 72-byte save area that has to be reserved by the problem program.

Return Codes in Register 15

00 Assignment successful.
04 No free LUB entry found.
08 Device not found in PUB table.
0C cuu has wrong device type.
10 cuu is down.
18 No free tape unit found.
1C Invalid logical unit for unassign.
20 cuu reserved by space management or by pending mount request.
24 Invalid function code.
28 No GETVIS space available.
2C Device to be unassigned is not assigned.
30 Device is owned by another partition.
34 Conflicting I/O assignment. Device is not assigned.
38 The specified logical unit number is invalid or not free.
3C No device with the specified mode was found.
40 No tape unit found which supports the specified mode.

ATTACH (Attach a Task) Macro



Requirements for the caller:

AMODE: 24 or 31

RMODE: 24 or ANY

ASC Mode: Primary

A subtask can be initiated by any other task of the partition with the ATTACH macro.

ATTACH supports the 31-bit environment as well as data spaces. ATTACH processing can attach a subtask in 24-bit or 31-bit addressing mode, physically resident above or below 16MB. When ATTACH is issued in 24-bit addressing mode, all operands are treated as 24-bit addresses. When ATTACH is issued in 31-bit addressing mode, all operands are treated as 31-bit addresses.

The attached task will get control in the same addressing mode as the issuer of the ATTACH macro. If, for example, a main task issues an ATTACH macro in AMODE 31, the subtask will also receive control in AMODE 31.

The maximum number of subtasks that can be initiated in the system at a time is 208. However, the maximum number of subtasks is also dependent on the number of partitions defined by the IPL SYS NPARTS command. Up to 31 subtasks can run concurrently within a partition, provided the overall limitation of 208 (or a lower number, dependent on NPARTS) is not exceeded.

If the maximum number of subtasks is already attached, any attempt to attach another subtask will be unsuccessful. This is indicated to the attaching task by a 1 in high-order bit 0 in register 1. Register 1 then points to an unposted ECB in the supervisor or the shared area (24 bit), and this ECB contains the reason code in byte 3. If byte 3 is zero, the maximum number of subtasks in the system is already attached or no system resources are available. A non-zero value indicates that the maximum number of 31 subtasks is already running in the partition. The attaching task may use this ECB to enter a wait state. The ECB is posted by the system whenever a task is available for attaching.

If the ATTACH macro successfully initiates a subtask, the attached task is given the lowest subtask priority, however, a higher priority than the main task. Register 1 of

the attached task contains the address of the attaching task's save area; the other registers contain the same values as those of the attaching task at the time when the ATTACH was issued. The address in register 1 can be used as the second operand of a POST macro later in the job if task-to-task communication is desired.

When SAVE is specified, register 0 of the attaching task contains the address of the byte immediately following the save area of the attached task, upon return from a successful ATTACH.

Note: If your program uses VSAM files, provide STXIT macros with AB and PC and issue a CLOSE or TCLOSE for the files before you cancel the subtask.

If register notation is used in any of the macro operands, register 0 and 1 should not be specified.

entrypoint | (S,entrypoint) | (r1)
The operand specifies the entrypoint of the subtask.

SAVE=savearea | (S,savearea) | (r2)
If specified, this operand must provide the address of the save area for the subtask. The save area is 120 bytes in length (=15 doublewords) and must be allocated below the 16MB line (RMODE 24).

If this operand is omitted, the supervisor allocates a save area for the attached subtask and passes its address in register 1 of the attaching task.

If an interrupt occurs while the subtask is in control, the system saves data in this area as follows (for the format of the area, see Figure 1):

The subtask's interrupt status information
The contents of the general purpose registers
The contents of the floating-point registers

Note: The status of the access registers will be saved in an internal save area.

Before issuing the ATTACH macro, move the subtask name in the first eight bytes of the save area. This name is used to identify the subtask if an abnormal end occurs.

Alternatively, you can specify the name of the subtask in the NAME operand of this macro.

Figure 1. Subtask-Save Area (120 Bytes)

Offset (In Hex)	Length (In Hex)	Length (In Dec)	Contents
0	8	8	Name of subtask.
8	8	8	Interrupt status.
10	40	64	Contents of registers 9 through 8 (one fullword per register).
50	8	8	Reserved.
58	20	32	Contents of floating-point registers

ABSAVE=absavearea | (S,absavearea) | (r3)

Specify this operand only if the subtask is to use the attaching task's abnormal termination routine (see the “STXIT (Set Exit) Macro” on page 383), that is, if it does not provide an abnormal termination routine of its own. The value specified in this operand must be the address of an AB exit save area for the subtask. If no AB exit is available, the specification is ignored.

If the ATTACH macro is issued in AMODE 31 or if the attaching task uses the extended save area layout (STXIT AMODE=ANY), the length of the AB exit save area must correspond to this layout. Otherwise the old STXIT save area is used. (See the “STXIT (Set Exit) Macro” on page 383 and the mapping “MAPSAVAR (Map Save Area) Macro” on page 309)

When an abnormal termination occurs, the supervisor saves the interrupt status and general registers 0 through 15 in this area before the exit is taken. In the extended save area, also the access registers are saved.

ECB=ecbname | (S,ecbname) | (r4)

Specify this operand if other tasks can be affected by this subtask's termination or if the ENQ and DEQ macros are used within the subtask. The operand is the name of the subtask's event control block (ECB). This block has a format as follows:

Bytes		Meaning of Bits if 1
0-1	Reserved	
2	0	Termination indicator
	1	Abnormal end indicator
	2-7	Reserved
3	Reserved	

When a subtask is attached, bits 0 and 1 of byte 2 are set to 0. When a subtask terminates, the supervisor sets byte 2, bit 0 of the ECB to 1. In addition, byte 2, bit 1 is set to 1 when the subtask ends abnormally; that is, if task termination is not caused by issuing one of the macros CANCEL, DETACH, DUMP, JDUMP, or EOJ.

NAME=name | (S,name) | (r5)

You can specify the subtask name here; however, only if you have omitted the SAVE operand (with the subtask name specification). It points to an eight-byte subtask name field.

If both the NAME and the SAVE operands are omitted, the supervisor allocates a save area for the subtask and provides a subtask name.

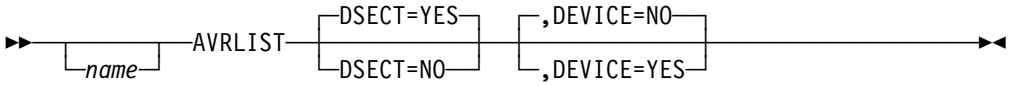
ALCOPY=YES | NO

This operand allows your program to transfer a copy of the attaching task's DU-AL to the subtask to be attached. In this way, the attaching program can share access to one or more data spaces with a program running under the subtask.

YES causes a copy of the caller's DU-AL to be given to the subtask. NO causes no access list to be given.

MFG=area | (S,area) | (r6)
The operand is required if the program which issues the ATTACH macro is to be reenterable. It specifies the address of a 64-byte storage area, that is, storage which your program may obtain through a GETVIS macro. This area is required for system use during execution of the macro.

AVRLIST (Map GETVCE) Macro



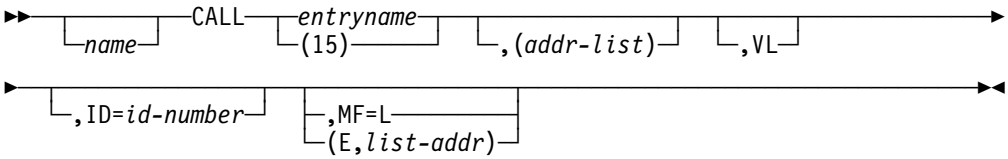
Required RMODE: **24**

The AVRLIST macro generates a DSECT describing volume characteristics retrieved with the GETVCE macro. (The DCTENTRY macro, called within AVRLIST if DEVICE=YES, describes device characteristics.)

DSECT=YES | NO
YES causes a mapping DSECT to be generated. NO causes inline code to be generated.

DEVICE=NO | YES
YES indicates that the macro DCTENTRY is to be called within AVRLIST, thus showing the complete (volume and device) output within one DSECT.

CALL (Call a Program) Macro



Requirements for the caller:

AMODE: 24 or 31

RMODE: 24 or ANY

ASC Mode: Primary or Access Register (AR)

Control parameters:
Must be in the caller's primary address space.

The CALL macro passes control from one program to a specified entry point in another program. You cannot use the CALL macro to pass control to a program in a different addressing mode; The AMODE of the caller is passed to the called program.

The CALL macro passes control to a control section at a specified entry point as follows: If a control section is not part of the object module which applies to the CALL macro, the linkage editor attempts to resolve this external reference by including the object module which contains the control section (AUTOLINK feature). When the CALL macro is executed, control is passed to the control section at the specified entry point.

The linkage relationship established when control is passed is the same as that created by a BAL instruction; that is, the issuing program expects control to be returned.

AR mode programs and primary mode programs can invoke the CALL macro. Before an AR mode program invokes this macro, the program must issue SYSSTATE ASCENV=AR to tell the CALL macro to generate code that is appropriate for AR mode.

entryname | (15)
Specifies the entry name to be given control.

(addr-list)
(addr-list),VL
Specifies one or more addresses (A-type address constants only), separated by commas, to be passed to the called program. To create the parameter list, the control program expands each address inline to a fullword on a fullword boundary in the specified order. Register 1 contains the address of the parameter list when the program receives control. (If this parameter is not coded, register 1 is not altered.)

VL is the default and causes the high-order bit of the last address parameter to be set to 1; the bit can be checked to find the end of the list.

If your program is in access register (AR) mode, the system builds the parameter list so that the addresses that are passed to the called program

are in the first half of the list and their associated ALETs are in the second half of the list. Therefore, the parameter list for callers in AR mode is twice as long as the parameter list for callers in primary mode for the same number of addresses. The 1 in the high-order bit identifies the last address parameter, but not the last entry in the parameter list.

ID=id-number

Specifies a 2-byte identifier useful for debugging purposes only. The last fullword of the macro expansion is a NOP instruction containing the identifier value in bytes 3 and 4.

MF=L | (E,list-addr)

L specifies the *list* form of the CALL macro, which generates a non-executable problem program parameter list that can be used by the execute form of the macro. In the list form, only A-type address constants may be used.

E specifies the *execute* form of the CALL macro, which uses the parameter list generated by the list form of the macro. **list-addr** specifies the address of the parameter list.

Only executable instructions and a VCON of the entry point are generated. If the address parameters are also specified in this form, the ADCONs of the parameter are placed on contiguous fullword boundaries beginning at the address specified in the MF parameter, and sequentially overlaying corresponding fullwords in the existing list.

CALL CSRPxxx (Call Cell Pool Services) Macro

The CALL CSRPxxx macro manages so-called cell pools, which are areas of virtual storage in address spaces or data spaces. A cell pool is subdivided into fixed-sized areas of storage called cells.

For detailed guide information on how to handle cell pool services, see the manual *VSE/ESA Extended Addressability* under “Callable Cell Pool Services”.

The CALL CSRPxxx macro supports the following main functions:

CSRPBLD
Build a cell pool

CSRPEXP
Expand a cell pool by adding an extent

CSRPCON
Connect cell storage to an extent

CSRPACT
Activate previously connected storage

CSRPDAC
Deactivate an extent

CSRPDIS
Disconnect the cell storage for an extent

CSRPGET and CSRPRGT
Allocate a cell from a cell pool

CSRPFRE and CSRPRFR
Return a cell to the cell pool

CSRPQPL
Query the cell pool

CSRPQEX
Query a cell pool extent

CSRPQCL
Query a cell

For a detailed description of these functions, see the “CALL (Call a Program)

Macro” on page 28 with the corresponding keyword (CALL CSRPBLD, CALL CSRPEXP,...).

Control Parameters

All parameters must reside in a single address or data space, and must be addressable by the caller. They must be in the primary address space or in an address/data space that is addressable through a public entry on the caller's dispatchable unit access list (DU-AL).

All variables must be A-type address constants.

Programming Requirements

If your program is in AR mode, issue the SYSSTATE macro with ASCENV=AR before you call the CSRPBLD service so the CALL macro can generate the correct code for AR mode.

Before you use cell pool services, you can optionally include the CSRCPASM macro to generate cell pool services equate (EQU) statements. CSRCPASM provides the following constants for use in your program:

```
*  Length of the cell pool anchor data area:
*
CSR_ANCHOR_LENGTH     EQU     64
*
*
*  Base length of the cell pool extent data area:
*
CSR_EXTENT_BASE       EQU     128
*
*
*  Length of the user-supplied pool name:
*
CSR_POOL_NAME_LEN     EQU     8
*
*
```

Register Information

Input

Before issuing a CALL CSRPxxx macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

Output

When control returns to the caller, the general purpose registers (GPRs) contain:

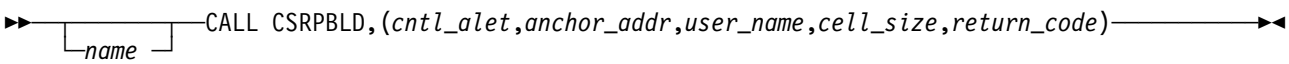
Register	Contents
0-1	Used by the system
2-13	Unchanged
14	Used by the system
15	Return code

When control returns to the caller, the access registers (ARs) contain:

Register	Contents
0-1	Used by the system
2-14	Unchanged
15	Used by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

CALL CSRPBLD (Build A Cell Pool And Initialize An Anchor)



Requirements for the caller:

AMODE: 24 or 31 (All input addresses must be valid 31-bit addresses.)

RMODE: 24 or ANY

ASC Mode: Primary or AR (If the anchor and the extents are located in a data space, the caller must be in AR mode.)

The CALL CSRPBLD cell pool service is used to format a 64-byte area for the cell pool anchor. You must first have acquired the storage for the anchor. You can call this service only once for a given cell pool.

cntl_alet Specifies the variable containing the ALET that identifies the location of the anchor and extents. Initialize the ALET to 0 if your program is running in AR mode and the anchor and extents are in the primary address space. If your program is running in primary ASC mode, the value is ignored, but you must code the parameter anyway.

anchor_addr Specifies the variable containing the address of the cell pool anchor.

user_name Specifies the 8-byte variable containing the name you want the service to assign to the pool. There are no restrictions on the name.

cell_size Specifies the variable containing the cell size in this pool. You can use any positive binary or hexadecimal number as the cell size.

return_code When CSRPBLD completes, this field (as well as R15) contains the return code.

Return Codes in Register 15

00 The operation was successful.

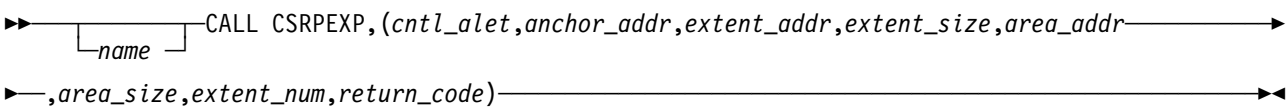
18 Program error. The anchor address is not valid.

Action: The upper address of the anchor exceeds the valid address range. Check to see if your program passed the wrong anchor address.

44 Program error. The cell size is not valid: it cannot be negative or 0.

Action: Specify a positive value for the cell size.

CALL CSRPEXP (Expand A Cell Pool)



Requirements for the caller:

AMODE: 24 or 31 (System code must be in 31-bit addressing mode when calling the service. All input addresses must be valid 31-bit addresses.)

RMODE: 24 or ANY

ASC Mode: Primary or AR (If the anchor and the extents are located in a data space, the caller must be in AR mode.)

The CALL CSRPEXP cell pool service is used to:

- Add an extent to the cell pool
- Assign a number to the extent
- Optionally, establish a connection between the extent and cell storage
- Optionally, make the cell storage available for allocation.

Note: If you are reusing an extent, use CSRPCON and CSRPACT instead of CSRPEXP.

If you specify zero for the cell storage size, CSRPEXP will add an extent to the cell pool, but will keep it in a disconnected state. When you specify the extent size, allow 128 bytes plus one byte per eight cells of cell storage. CSRPEXP allocates cells contiguously, starting at the address you specify. If you specify zero for the area length, CSRPEXP ignores the area address.

cntl_alet Specifies the variable containing the ALET that identifies the location of the anchor and extents. Initialize the ALET to 0 if your program is running in AR mode and the anchor and extents are in the primary address space. If your program is running in primary ASC mode, the value is ignored, but you must code the parameter anyway.

anchor_addr Specifies the variable containing the address of the 64-byte anchor.

extent_addr Specifies the variable containing the address of the extent.

extent_size Specifies the variable containing the size of the extent.

area_addr Specifies the variable containing the starting address of the cell storage area. The starting address of this area must be consistent with any boundary requirements that you might have.

area_size Specifies the variable containing the size (binary or hexadecimal) of the storage area for the cells.

extent_num When CSRPEXP completes, the variable specifying *extent_num* contains the number of the extent to be connected. You will use this number on subsequent CALLs.

return_code When CSRPEXP completes, the variable specifying *return_code* contains the return code.

Return Codes in Register 15

00 The operation was successful.

0C Program error. There are too many extents in the cell pool.

Action: Check to see if your program contains a logic error that caused the limit of 65,536 extents per cell pool to be exceeded. If your program works as expected, consider using a larger cell pool.

1C Program error. The anchor address is not valid.

Action: Check to see if your program passed the wrong anchor address or inadvertently overlaid the anchor area.

28 Program error. The service could not use the extent address.

Action: Make sure that the extent area does not overlap the anchor area. Also make sure that the upper address of the extent does not exceed the valid address range.

2C Program error. The extent length is not valid.

Action: Correct the extent length. It cannot be less than 129 bytes.

48 Program error. The cell area length is not valid.

Action: Correct the cell area length. The cell area size cannot be less than the cell size.

4C Program error. The service could not use the cell area address.

Action: If the cell area is in a data space, make sure the cell area is completely within the data space.

50 Program error. The cell area is too large.

Action: Specify a larger extent size or a smaller cell area size.

64 Program error or system error. An extent chain was broken.

Action: Check to see if your program inadvertently overlaid an extent area. Make sure that no extent belongs to more than one cell pool.

68 Program error or system error. An extent chain is circular.

Action: Check to see if your program inadvertently overlaid an extent area. Make sure that no extent belongs to more than one cell pool.

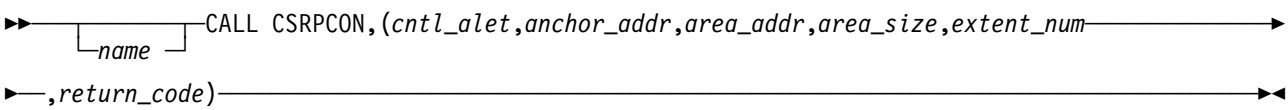
70 Program error or system error. An anchor has been overlaid.

Action: Check to see if your program inadvertently overlaid the anchor area.

74 Program error or system error. An extent has been overlaid.

Action: Check to see if your program inadvertently overlaid an extent area. Make sure that no extent belongs to more than one cell pool.

CALL CSRPCON (Connect Cell Storage to an Extent)



Requirements for the caller:

AMODE: 24 or 31 (System code must be in 31-bit addressing mode when calling the service. All input addresses must be valid 31-bit addresses.)

RMODE: 24 or ANY

ASC Mode: Primary or AR (If the anchor and the extents are located in a data space, the caller must be in AR mode.)

The CALL CSRPCON cell pool service is used to connect cell storage to the extent that you specify or to reuse a disconnected extent. The CSRPEXP service returned the extent number. The extent must be in the disconnected state, which means that you have not called CSRPACT to activate this particular extent.

cntl_alet Specifies the variable containing the ALET that identifies the location of the anchor and extents. Initialize the ALET to 0 if your program is running in AR mode and the anchor and extents are in the primary address space. If your program is running in primary ASC mode, the value is ignored, but you must code the parameter anyway.

anchor_addr Specifies the variable containing the address of the 64-byte anchor.

area_addr Specifies the variable containing the starting address of the cell storage area. The starting address of this area must be consistent with any boundary requirements that you might have.

area_size Specifies the variable containing the size (binary or hexadecimal) of the storage area for the cells. CSRPCON determines the number of cells that will fit in the area.

extent_num When CSRPCON completes, the variable specifying *extent_num* contains the number of the extent to be connected. The extent number must be within the range 0 to 65,536.

return_code When CSRPCON completes, the variable specifying *return_code* contains the return code.

Return Codes in Register 15

00 The operation was successful.

1C Program error. The anchor address is not valid.

Action: Check to see if your program passed the wrong anchor address or inadvertently overlaid the anchor area.

30 Program error. The extent number is not valid.

Action: Specify the extent number within the range 1 to 65,536.

34 Program error. You issued the services in the wrong order, or did not issue a necessary service.

Action: Check to see if your program passed the wrong extent number. Make sure that the extent is in a disconnected state (that is, it has not been activated through CSRPACT or CSRPEXP).

48 Program error. The cell area length is not valid.

Action: Correct the cell area length. The cell area size cannot be less than the cell size.

4C Program error. The service could not use the cell area address.

Action: If the cell area is in a data space, make sure the cell area is completely within the data space.

50 Program error. The cell area is too large.

Action: Specify a larger extent size or a smaller cell area size.

64 Program error or system error. An extent chain was broken.

Action: Check to see if your program inadvertently overlaid an extent area. Make sure that no extent belongs to more than one cell pool.

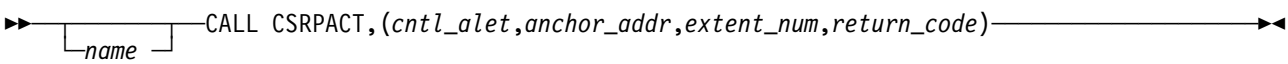
68 Program error or system error. An extent chain is circular.

Action: Check to see if your program inadvertently overlaid an extent area. Make sure that no extent belongs to more than one cell pool.

6C Program error or system error. An extent could not be found.

Action: Check to see if your program inadvertently overlaid an extent area. Make sure that the anchor address being passed is for the right cell pool.

CALL CSRPACT (Activate Previously Connected Storage)



Requirements for the caller:

AMODE: 24 or 31 (System code must be in 31-bit addressing mode when calling the service. All input addresses must be valid 31-bit addresses.)

RMODE: 24 or ANY

ASC Mode: Primary or AR (If the anchor and the extents are located in a data space, the caller must be in AR mode.)

The CALL CSRPACT cell pool service is used to activate the extent cell storage for allocation. You must specify which extent you want to activate.

cntl_alet Specifies the variable containing the ALET that identifies the location of the anchor and extents. Initialize the ALET to 0 if your program is running in AR mode and the anchor and extents are in the primary address space. If your program is running in primary ASC mode, the value is ignored, but you must code the parameter anyway.

anchor_addr Specifies the variable containing the address of the 64-byte anchor.

extent_num Specifies the variable containing the number of the extent to be connected. The extent number must be within the range 0 to 65,536.

return_code When CSRPACT completes, the variable specifying *return_code* contains the return code.

Return Codes in Register 15

00 The operation was successful.

1C Program error. The anchor address is not valid.

Action: Check to see if your program passed the wrong anchor address or inadvertently overlaid the anchor area.

30 Program error. The extent number is not valid.

Action: Specify the extent number within the range 1 to 65,536.

34 Program error. The extent is in an incorrect state.

Action: Check to see if your program passed the wrong extent number. Make sure that the extent is not already in an active state (that is, it has not been activated through CSRPACT or CSRPEXP). Also make sure that the extent is not in a disconnected state.

64 Program error or system error. An extent chain was broken.

Action: Check to see if your program inadvertently overlaid an extent area. Make sure that no extent belongs to more than one cell pool.

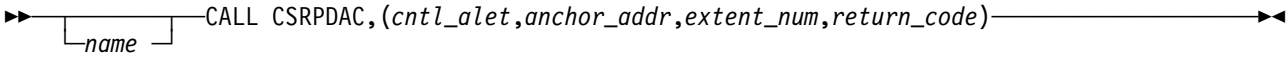
68 Program error or system error. An extent chain is circular.

Action: Check to see if your program inadvertently overlaid an extent area. Make sure that no extent belongs to more than one cell pool.

6C Program error or system error. An extent could not be found.

Action: Check to see if your program inadvertently overlaid an extent area. Make sure that the anchor address being passed is for the right cell pool.

CALL CSRPDAC (Deactivate an Extent)



Requirements for the caller:

AMODE: 24 or 31 (System code must be in 31-bit addressing mode when calling the service. All input addresses must be valid 31-bit addresses.)

RMODE: 24 or ANY

ASC Mode: Primary or AR (If the anchor and the extents are located in a data space, the caller must be in AR mode.)

The CALL CSRPDAC cell pool service is used to deactivate a specific extent. You must specify which extent you want to deactivate.

cntl_alet Specifies the variable containing the ALET that identifies the location of the anchor and extents. Initialize the ALET to 0 if your program is running in AR mode and the anchor and extents are in the primary address space. If your program is running in primary ASC mode, the value is ignored, but you must code the parameter anyway.

anchor_addr Specifies the variable containing the address of the 64-byte anchor.

extent_num Specifies the variable containing the number of the extent to be disconnected. The extent number must be within the range 0 to 65,536.

return_code When CSRPDAC completes, the variable specifying *return_code* contains the return code.

Return Codes in Register 15

00 The extent has been deactivated, but there are still cells allocated.

04 The extent has been deactivated and there are no allocated cells remaining.

Action: None required.

1C Program error. The anchor address is not valid.

Action: Check to see if your program passed the wrong anchor address or inadvertently overlaid the anchor area.

30 Program error. The extent number is not valid.

Action: Specify the extent number within the range 1 to 65,536.

34 Program error. You issued the services in the wrong order or did not issue a necessary service.

Action: Check to see if your program passed the wrong extent number. Make sure that the extent is in an active state before calling the service.

64 Program error or system error. An extent chain was broken.

Action: Check to see if your program inadvertently overlaid an extent area. Make sure that no extent belongs to more than one cell pool.

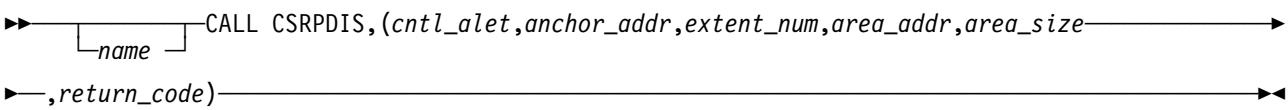
68 Program error or system error. An extent chain is circular.

Action: Check to see if your program inadvertently overlaid an extent area. Make sure that no extent belongs to more than one cell pool.

6C Program error or system error. An extent could not be found.

Action: Check to see if your program inadvertently overlaid an extent area. Make sure that the anchor address being passed is for the right cell pool.

CALL CSRPDIS (Disconnect the Cell Storage for an Extent)



Requirements for the caller:

AMODE: 24 or 31 (System code must be in 31-bit addressing mode when calling the service. All input addresses must be valid 31-bit addresses.)

RMODE: 24 or ANY

ASC Mode: Primary or AR (If the anchor and the extents are located in a data space, the caller must be in AR mode.)

The CALL CSRPDIS cell pool service is used to disconnect cell storage for a specific extent.

cntl_alet Specifies the variable containing the ALET that identifies the location of the anchor and extents. Initialize the ALET to 0 if your program is running in AR mode and the anchor and extents are in the primary address space. If your program is running in primary ASC mode, the value is ignored, but you must code the parameter anyway.

anchor_addr Specifies the variable containing the address of the 64-byte anchor.

extent_num Specifies the variable containing the number of the extent to be disconnected. The extent number must be within the range 1 to 65,536.

area_addr When CSRPDIS completes, the variable specifying *area_addr* contains the address of the disconnected storage area.

area_size When CSRPDIS completes, the variable specifying *area_size* contains the size of the disconnected storage area.

return_code When CSRPDIS completes, the variable specifying *return_code* contains the return code.

Return Codes in Register 15

00 The operation was successful.

1C Program error. The anchor address is not valid.

Action: Check to see if your program passed the wrong anchor address or inadvertently overlaid the anchor area.

30 Program error. The extent number is not valid.

Action: Specify the extent number within the range 0 to 65,536.

34 Program error. You issued the services in the wrong order or did not issue a necessary service.

Action: Call CSRPDAC to deactivate the extent before calling CSRPDIS to disconnect the cell storage for the extent.

38 Program error. The service cannot disconnect the extent because some cells are still allocated.

Action: Return all the cells associated with the extent before calling CSRPDIS to disconnect the cell storage for the extent.

64 Program error or system error. An extent chain was broken.

Action: Check to see if your program inadvertently overlaid an extent area. Make sure that no extent belongs to more than one cell pool.

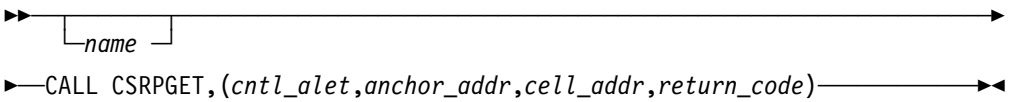
68 Program error or system error. An extent chain is circular.

Action: Check to see if your program inadvertently overlaid an extent area. Make sure that no extent belongs to more than one cell pool.

6C Program error or system error. An extent could not be found.

Action: Check to see if your program inadvertently overlaid an extent area. Make sure that the anchor address being passed is for the right cell pool.

CALL CSRPGET (Allocate a Cell from a Cell Pool)



Requirements for the caller:

AMODE: 24 or 31 (System code must be in 31-bit addressing mode when calling the service. All input addresses must be valid 31-bit addresses.)

RMODE: 24 or ANY

ASC Mode: Primary or AR (If the anchor and the extents are located in a data space, the caller must be in AR mode.)

The CALL CSRPGET cell pool service is used to allocate a cell from the cell pool. CSRPGET allocates cells from the lowest- to the highest-numbered active extents and - within each extent - from the lowest to the highest cell address. CSRPGET passes back to the calling program the address of the cell it allocated, but does not clear the cell storage to binary zeros.

cntl_alet Specifies the variable containing the ALET that identifies the location of the anchor and extents. Initialize the ALET to 0 if your program is running in AR mode and the anchor and extents are in the primary address space. If your program is running in primary ASC mode, the value is ignored, but you must code the parameter anyway.

anchor_addr Specifies the variable containing the address of the 64-byte anchor.

cell_addr When CSRPGET completes, the variable specifying *cell_addr* contains the address of the cell that CSRPGET allocated.

return_code When CSRPGET completes, the variable specifying *return_code* contains the return code.

Return Codes in Register 15

00 The operation was successful.

08 Program error. There were no available cells in the pool. More than one program could be using the cell pool.

Action: Retry the request one or more times. If the problem persists, consider freeing existing cells or adding new cells to the cell pool, or both.

1C Program error. The anchor address is not valid.

Action: Check to see if your program passed the wrong anchor address or inadvertently overlaid the anchor area.

64 Program error or system error. An extent chain was broken.

Action: Check to see if your program inadvertently overlaid an extent area. Make sure that no extent belongs to more than one cell pool.

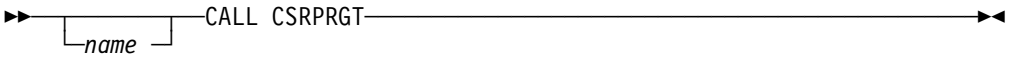
68 Program error or system error. An extent chain is circular.

Action: Check to see if your program inadvertently overlaid an extent area. Make sure that no extent belongs to more than one cell pool.

74 Program error or system error. An extent has been overlaid.

Action: Check to see if your program inadvertently overlaid an extent area. Make sure that no extent belongs to more than one cell pool.

CALL CSRPRGT (Allocate a Cell from a Cell Pool - Register Interface)



Requirements for the caller:

AMODE: 24 or 31 (System code must be in 31-bit addressing mode when calling the service. All input addresses must be valid 31-bit addresses.)

RMODE: 24 or ANY

ASC Mode: Primary or AR (If the anchor and the extents are located in a data space, the caller must be in AR mode.)

The CALL CSRPRGT cell pool service is used to allocate a cell from the cell pool using the register interface (in case your program cannot obtain storage for a parameter list). CSRPRGT allocates cells from the lowest- to the highest-numbered active extents and - within each extent - from the lowest to the highest cell address.

Input Register Information

Before calling the CSRPRGT service, the caller must ensure that the following access registers (ARs) and general purpose registers (GPRs) contain the specified information:

Register	Contents
AR 1	The ALET used to access all the cell storage areas. Specify 0 if your program is running in AR mode and the anchor and extents are in the primary address space. If your program is running in primary ASC mode, CSRPRGT ignores the value.
GPR 1	The anchor address

Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Used as a work register by the system
1	Address of the allocated cell
2-13	Unchanged
14	Used as a work register by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0	Used as a work register by the system
1-14	Unchanged
15	Used as a work register by the system

Return Codes in Register 15

00 The operation was successful.

08 Program error. There were no available cells in the pool.

Action: Retry the request one or more times. If the problem persists, consider freeing existing cells or adding new cells to the cell pool, or both.

1C Program error. The anchor address is not valid.

Action: Check to see if your program passed the wrong anchor address or inadvertently overlaid the anchor area.

64 Program error or system error. An extent chain was broken.

Action: Check to see if your program inadvertently overlaid an extent area. Make sure that no extent belongs to more than one cell pool.

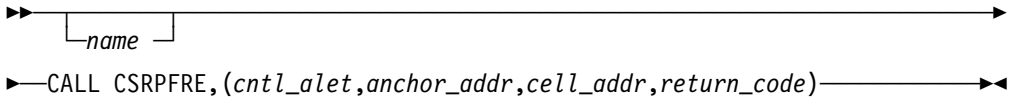
68 Program error or system error. An extent chain is circular.

Action: Check to see if your program inadvertently overlaid an extent area. Make sure that no extent belongs to more than one cell pool.

74 Program error or system error. An extent has been overlaid.

Action: Check to see if your program inadvertently overlaid an extent area. Make sure that no extent belongs to more than one cell pool.

CALL CSRPFRE (Return a Cell to a Cell Pool)



Requirements for the caller:

AMODE: 24 or 31 (System code must be in 31-bit addressing mode when calling the service. All input addresses must be valid 31-bit addresses.)

RMODE: 24 or ANY

ASC Mode: Primary or AR (If the anchor and the extents are located in a data space, the caller must be in AR mode.)

The CALL CSRPFRE cell pool service is used to return an allocated cell to the cell pool. You must specify the address of the cell that you want to return.

cntl_alet Specifies the variable containing the ALET that identifies the location of the anchor and extents. Initialize the ALET to 0 if your program is running in AR mode and the anchor and extents are in the primary address space. If your program is running in primary ASC mode, the value is ignored, but you must code the parameter anyway.

anchor_addr Specifies the variable containing the address of the 64-byte anchor.

cell_addr Specifies the variable containing the address of the cell that CSRPFRE is to free.

return_code When CSRPFRE completes, the variable specifying *return_code* contains the return code.

Return Codes in Register 15

00 The operation was successful.

04 The last cell has been returned to an inactive extent.

Action: None required. However, you might take some action depending on your application.

1C Program error. The anchor address is not valid.

Action: Check to see if your program passed the wrong anchor address or inadvertently overlaid the anchor area.

54 Program error. The cell address is not valid.

Action: Investigate the following possible causes:

- The input cell address does not point to the beginning of a cell.
- The cell is not in the cell pool specified by the anchor address.

58 Program error. Either you have already returned the cell or you never allocated it.

Action: Check to see if your program contains a logic error that caused this situation to occur.

64 Program error or system error. An extent chain was broken.

Action: Check to see if your program inadvertently overlaid an extent area. Make sure that no extent belongs to more than one cell pool.

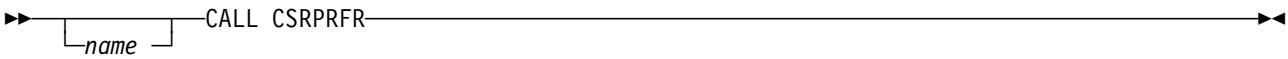
68 Program error or system error. An extent chain is circular.

Action: Check to see if your program inadvertently overlaid an extent area. Make sure that no extent belongs to more than one cell pool.

74 Program error or system error. An extent has been overlaid.

Action: Check to see if your program inadvertently overlaid an extent area. Make sure that no extent belongs to more than one cell pool.

CALL CSRPRFR (Return a Cell to a Cell Pool - Register Interface)



Requirements for the caller:

AMODE: 24 or 31 (System code must be in 31-bit addressing mode when calling the service. All input addresses must be valid 31-bit addresses.)

RMODE: 24 or ANY

ASC Mode: Primary or AR (If the anchor and the extents are located in a data space, the caller must be in AR mode.)

The CALL CSRPRFR cell pool service is used to return an allocated cell to the cell pool using the register interface (in case your program cannot obtain storage for a parameter list).

Input Register Information

Before calling the CSRPRFR service, the caller must ensure that the following access registers (ARs) and general purpose registers (GPRs) contain the specified information:

Register	Contents
AR 1	The ALET used to access all the cell storage areas. Specify 0 if your program is running in AR mode and the anchor and extents are in the primary address space. If your program is running in primary ASC mode, CSRPRFR ignores the value.
GPR 0	The address of the cell you want to be freed.
GPR 1	The anchor address.

Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0-1	Used as a work registers by the system
2-13	Unchanged
14	Used as a work register by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as a work registers by the system
2-14	Unchanged
15	Used as a work register by the system

Return Codes in Register 15

00 The operation was successful.

04 The last cell has been returned to an inactive extent.

Action: None required. However, you might want to take some action depending on your application.

1C Program error. The anchor address is not valid.

Action: Check to see if your program passed the wrong anchor address or inadvertently overlaid the anchor area.

54 Program error. The cell address is not valid.

Action: Investigate the following possible causes:

- The input cell address does not point to the beginning of a cell.
- The cell is not in the cell pool specified by the anchor address.

58 Program error. Either you have already returned the cell or you never allocated it.

Action: Check to see if your program contains a logic error that caused this situation to occur.

64 Program error or system error. An extent chain was broken.

Action: Check to see if your program inadvertently overlaid an extent area. Make sure that no extent belongs to more than one cell pool.

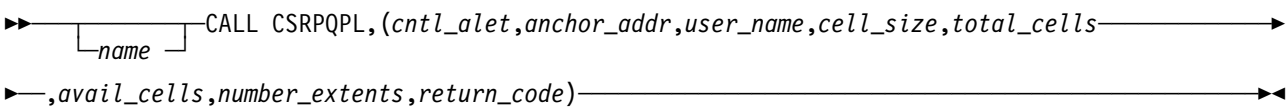
68 Program error or system error. An extent chain is circular.

Action: Check to see if your program inadvertently overlaid an extent area. Make sure that no extent belongs to more than one cell pool.

74 Program error or system error. An extent has been overlaid.

Action: Check to see if your program inadvertently overlaid an extent area. Make sure that no extent belongs to more than one cell pool.

CALL CSRPQPL (Query the Cell Pool)



Requirements for the caller:

AMODE: 24 or 31 (System code must be in 31-bit addressing mode when calling the service. All input addresses must be valid 31-bit addresses.)

RMODE: 24 or ANY

ASC Mode: Primary or AR (If the anchor and the extents are located in a data space, the caller must be in AR mode.)

The CALL CSRPQPL cell pool service is used to receive status information about the cell pool. CSRPQPL does not prevent other programs from changing the pool during or after a query. CSRPQPL returns the status as it was at the time you issued the call.

cntl_alet Specifies the variable containing the ALET that identifies the location of the anchor and extents. Initialize the ALET to 0 if your program is running in AR mode and the anchor and extents are in the primary address space. If your program is running in primary ASC mode, the value is ignored, but you must code the parameter anyway.

anchor_addr Specifies the variable containing the address of the 64-byte anchor.

user_name When CSRPQPL completes, the variable specified by *user_name* contains the name on the CSRPBLD service that created the cell pool.

cell_size When CSRPQPL completes, the variable specified by *cell_size* contains the size of each cell at the time the cell pool was created.

total_cells When CSRPQPL completes, the variable specified by *total_cells* contains the total number of cells associated with the extent.

avail_cells When CSRPQPL completes, the variable specified by *avail_cells* contains the total number of cells in active extents that are available for allocation.

number_extents When CSRPQPL completes, the variable specified by *number_extents* contains the total number of extents (active or inactive, connected or disconnected) in the cell pool.

return_code When CSRPFRE completes, the variable specifying *return_code* contains the return code.

Return Codes in Register 15

00 The operation was successful.

1C Program error. The anchor address is not valid.

Action: Check to see if your program passed the wrong anchor address or inadvertently overlaid the anchor area.

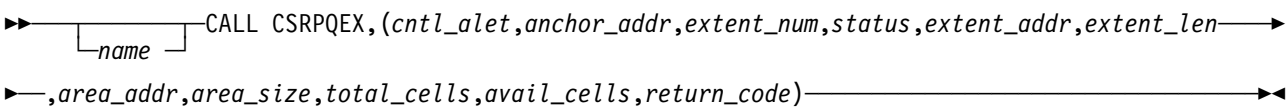
64 Program error or system error. The extent address is not valid.

Action: Check to see if your program inadvertently overlaid an extent area. Make sure that no extent belongs to more than one cell pool.

68 Program error or system error. An extent chain is circular.

Action: Check to see if your program inadvertently overlaid an extent area. Make sure that no extent belongs to more than one cell pool.

CALL CSRPQEX (Query a Cell Pool Extent)



Requirements for the caller:

AMODE: 24 or 31 (System code must be in 31-bit addressing mode when calling the service. All input addresses must be valid 31-bit addresses.)

RMODE: 24 or ANY

ASC Mode: Primary or AR (If the anchor and the extents are located in a data space, the caller must be in AR mode.)

The CALL CSRPQEX cell pool service is used to receive status information about a specified extent.

cntl_alet Specifies the variable containing the ALET that identifies the location of the anchor and extents. Initialize the ALET to 0 if your program is running in AR mode and the anchor and extents are in the primary address space. If your program is running in primary ASC mode, the value is ignored, but you must code the parameter anyway.

anchor_addr Specifies the variable containing the address of the 64-byte anchor.

extent_num Specifies the variable containing the number of the extent the service will query.

status When CSRPQEX completes, the variable specified for *status* contains one of the following decimal numbers. These indicate the status of the extent at the time of the CALL.

1 Disconnected and inactive
2 Connect in progress
3 Connected and inactive
4 Connected and active
5 Disconnect in progress

extent_addr When CSRPQEX completes, the variable specified for *extent_addr* contains the address of the extent.

extent_len When CSRPQEX completes, the variable specified for *extent_len* contains the length of the extent, in bytes.

area_addr When CSRPQEX completes, the variable specified for *area_addr* contains the address of cell storage.

,area_size When CSRPQEX completes, the variable specified for *area_size* contains the size of cell storage for the extent.

total_cells When CSRPQEX completes, the variable specified by *total_cells* contains the total number of cells associated with the extent.

avail_cells When CSRPQEX completes, the variable specified by *avail_cells* contains the total number of cells in active extents that are available for allocation.

number_extents When CSRPQEX completes, the variable specified by *number_extents* contains the total number of extents (active or inactive, connected or disconnected) in the cell pool.

return_code When CSRPQEX completes, the variable specifying *return_code* contains the return code.

Return Codes in Register 15

00 The operation was successful.

1C Program error. The anchor address is not valid.

Action: Check to see if your program passed the wrong anchor address or inadvertently overlaid the anchor area.

30 Program error. The extent number is not valid.

Action: Specify the extent number within the range 1 to 65,536.

64 Program error or system error. The extent address is not valid.

Action: Check to see if your program inadvertently overlaid an extent area. Make sure that no extent belongs to more than one cell pool.

68 Program error or system error. An extent chain is circular.

Action: Check to see if your program inadvertently overlaid an extent area. Make sure that no extent belongs to more than one cell pool.

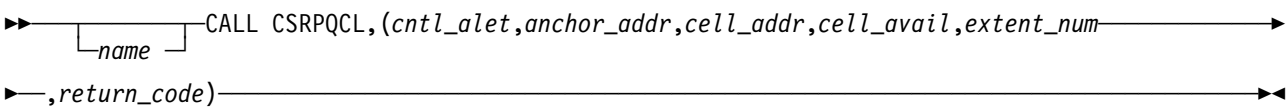
6C Program error or system error. An extent could not be found.

Action: Check to see if your program inadvertently overlaid an extent area. Make sure that no extent belongs to more than one cell pool.

74 Program error or system error. An extent has been overlaid.

Action: Check to see if your program inadvertently overlaid an extent area. Make sure that no extent belongs to more than one cell pool.

CALL CSRPQCL (Query a Cell)



Requirements for the caller:

AMODE: 24 or 31 (System code must be in 31-bit addressing mode when calling the service. All input addresses must be valid 31-bit addresses.)

RMODE: 24 or ANY

ASC Mode: Primary or AR (If the anchor and the extents are located in a data space, the caller must be in AR mode.)

The CALL CSRPQCL cell pool service is used to receive status information about a specified cell in a cell pool. CSRPQCL reports whether the cell is free or allocated, and returns the number of the extent associated with the cell. CSRPQCL does not prevent other programs from changing the pool during or after a query. CSRPQCL returns the status as it was at the time you issued the call.

cntl_alet Specifies the variable containing the ALET that identifies the location of the anchor and extents. Initialize the ALET to 0 if your program is running in AR mode and the anchor and extents are in the primary address space. If your program is running in primary ASC mode, the value is ignored, but you must code the parameter anyway.

anchor_addr Specifies the variable containing the address of the 64-byte anchor.

cell_addr Specifies the variable containing the address of the cell to be queried.

cell_avail When CSRPQCL completes, the variable specified for *cell_avail* contains one of the following values. These indicate the status of the specified cell at the time you issued the CALL macro.

- **0** Cell available
- **1** Cell allocated

extent_num When CSRPQCL completes, the variable specified for *extent_num* contains the number of the extent that contains the specified cell.

return_code When CSRPQCL completes, the variable specifying *return_code* contains the return code.

Return Codes in Register 15

00 The operation was successful.

1C Program error. The anchor address is not valid.

Action: Check to see if your program passed the wrong anchor address or inadvertently overlaid the anchor area.

54 Program error. The cell address is not valid.

Action: Investigate the following possible causes:

- The input cell address does not point to the beginning of a cell.
- The cell is not in the cell pool specified by the anchor address.

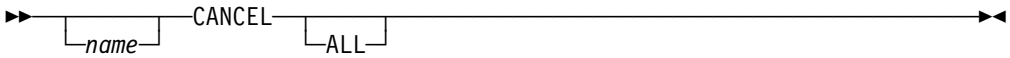
64 Program error or system error. An extent chain was broken.

Action: Check to see if your program inadvertently overlaid an extent area. Make sure that no extent belongs to more than one cell pool.

68 Program error or system error. An extent chain is circular.

Action: Check to see if your program inadvertently overlaid an extent area. Make sure that no extent belongs to more than one cell pool.

CANCEL (Cancel Task) Macro



Requirements for the caller:

AMODE: 24 or 31

RMODE: 24 or ANY

ASC Mode: Primary

Issuing the CANCEL macro in a subtask abnormally terminates the subtask without branching to any abnormal termination routine.

A CANCEL ALL macro issued in a subtask, or a CANCEL issued in the main task, abnormally terminates all processing in the partition (job). Job termination in multitasking causes all abnormal termination exits (via STXIT AB) to be taken for each task except the one that issued the CANCEL macro. Once these exits are taken, the job is terminated. System messages (using the subtask name) are issued to identify each of the terminated subtasks.

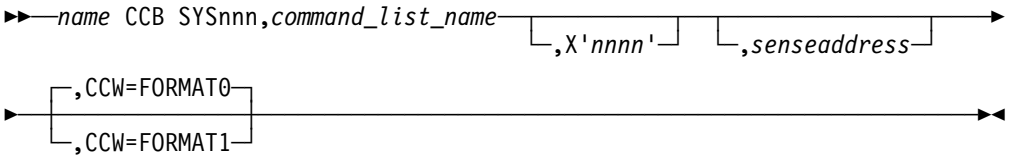
If the CANCEL macro is issued without an operand, you may not code a comment unless this comment begins with a comma. If CANCEL ALL is issued, you may code a comment as usual.

If the DUMP option was specified and SYSLST is assigned, a system dump will occur

- if a CANCEL ALL macro is issued by a subtask, or
- if a CANCEL macro is issued by a main task with subtasks attached.

Note: If your program uses VSAM files, ensure that these files are closed before you issue this macro.

CCB (Command Control Block Definition) Macro



Required RMODE: **24**

A CCB (command control block) macro must be specified in your program for each I/O device controlled by physical IOCS macros. The CCB (see Figure 2 on page 57) is needed to communicate information to physical IOCS so that it can perform desired operations (for example, indicating printer channel 9). The CCB also receives status information after an operation and makes this available to your program. You should ensure proper boundary alignment of the CCB if this is necessary for your program.

Note: In some applications, it may be preferable to use an IORB (I/O Request Block) in place of a CCB. Do this by specifying either an IORB or GENIORB macro.

name The CCB macro must be given a symbolic name (blockname). This name can be used as the operand in the EXCP and WAIT macros which refer to the CCB.

SYSnnn
This operand specifies the symbolic unit for the actual I/O unit with which this CCB is associated. The actual I/O unit can be assigned to the symbolic unit by an ASSGN job control statement.

command_list_name
This operand specifies the symbolic name of the first CCW used with a CCB. This name must be the same as the name specified in the assembler CCW statement that constructs the CCW.

X'nnnn'
A hexadecimal value used to set the CCB user option bits. For the value to be set, if applicable, see the section “CCB Communication Bytes” on page 58. If more than one bit must be set, use the sum of the values.

senseaddress
This operand, when supplied, indicates user error recovery (bit 7 of byte 2 must be on – for more details about this bit, see page on page 59) and generates a CCW for reading sense information (as the last field of the CCB). The name field (sense address) of the area that you supply must have a length attribute assigned of at least one byte. Physical IOCS uses this length attribute in the CCW to determine the number of bytes of sense information you desire.

CCW=FORMAT0 | FORMAT1
Indicates whether format-0 CCWs (for I/O areas below 16MB) or format-1 CCWs (for I/O areas above 16MB) are used. (You can use the High-Level Assembler (HLASM) instruction CCW1 to generate a format-1 CCW.)

The format-1 CCW is invalid for EXCP REAL (user-translated CCB in virtual partition).

Format of the CCB

The macro sets up an area of either 16 bytes or 24 bytes. For the layout of this area and its contents see Figure 2 and the following description of the individual fields.

Figure 2. Layout and Contents of the Command Control Block (CCB)

Offset (In Hex)	Length (In Hex)	Contents
0	2	Residual count (see Note 1).
2	2	Communication bytes (see Note 2).
4	2	CSW status bits (see Note 3).
6	2	Type code (see Note 4 on page 58).
8	1	Reserved.
9	3	Address of CCW (see Note 5 on page 58).
0C	1	Reserved.
0D	3	CCW address in CSW (see Note 6 on page 58).
10	10	Optional sense CCW (see Note 7 on page 58).

Notes:

1. After a record has been transferred, IOCS places the residual count from the CSW into bytes 0 and 1. By subtracting this count from the original count in the CCW, your program can determine the length of the transferred record.
2. The two bytes, also known as transmission information, are used for communicating information between physical IOCS and your program. For detailed information on the use and purpose of the bits in this field, see the section “CCB Communication Bytes” on page 58.

   Your program can test any of the bits in this field using the mask given for each of the bits. Your program may test more than one bit by the hexadecimal sum of the test values.

   All bits are set to 0 when your program is assembled unless the X'nnnn' operand is specified. If this operand is specified, it is assembled into these two bytes. When your program is being executed, a bit may be set to 1 by your program (to request certain functions or specific feedback information) or by physical IOCS (as a result of having detected a certain condition). Any bits that can be turned on by physical IOCS during program execution are reset to zero by PIOCS the next time an EXCP macro is executed against the same CCB.
3. Bytes 4 and 5 are set to X'00' whenever an EXCP macro is issued against the CCB. The meaning of the bits in these two bytes is as follows:

```
Byte 4:                    Byte 5:

0 = Attention              0 = Program-controlled interruption
1 = Status modifier        1 = Incorrect length
2 = Control unit end       2 = Program check
3 = Busy                   3 = Protection check
4 = Channel end            4 = Channel data check
5 = Device end             5 = Channel control check
6 = Unit check             6 = Interface control check
7 = Unit exception         7 = Chaining check
```

4. Contents of byte 6:

   ```
   X'0u' = Original CCB
   X'4u' = BTAM-ES CCB
   X'8u' = User-translated CCB in virtual partition

   If u = 0: the address in byte 7 refers to a system
             logical unit.
   If u = 1: the address in byte 7 refers to a programmer
             logical unit.
   ```

   Contents of byte 7: Hexadecimal representation of SYSnnn:

   ```
   SYSRDR = 00        SYS000 = 00
   SYSIPT = 01        SYS001 = 01
   SYSPCH = 02        SYS002 = 02
   SYSLST = 03        .
   SYSLOG = 04        .
   SYSLNK = 05        .
   SYSRES = 06        SYS254 = FE
   SYSUSE = 09
   SYSREC = 0A
   SYSCAT = 0D
   ```

5. Bytes 9 through 11 contain the address of the CCW (or of the first of a chain of CCWs) associated with the CCB:

   This is a real address if CCB byte 6 = X'8u'.
   This is a virtual address if CCB byte 6 = X'0u'.

6. Bytes 13-15 contain either of the following:

   The CCW address contained in the CSW at channel-end interrupt for the I/O operation involving the CCB; or the address of the associated channel appendage routine if CCB byte 12 contains X'40'.

7. Bytes 16 to 23 are provided only if the sense operand was specified in the CCB macro. They accommodate the CCW for returning sense information to your program.

CCB Communication Bytes

CCB Byte 2

Bit 0 – Traffic bit:
If 0 I/O requested and not completed.
If 1 I/O completed. Normally set at channel end. Set at device end if bit 5 is 1.
Bit 1 – End-of-file on system input:

If 1 /* or /& on SYSRDR or SYSIPT. Bit 7 of byte 4 (unit exception) is also on.

For a PRT1 printer (see also Note 1 on page 61) – a UCB parity check (line complete).

Bit 2 – Irrecoverable I/O error:

If 0 No program- or operator-option error was passed back.

If 1 I/O error was passed back due to a program or an operator option.

Bit 3 – Accept irrecoverable I/O error – Bit 2 is set to 1 (see also Note 2 on page 61):

If 0 Cancel in case of permanent I/O error.

If 1 Return to the user in case of permanent I/O error.

ON value for the third operand of the CCB macro: X'1000'.

Bit 4 – Return:

Disk data check
Data check on an IBM 3540
Indicate A-type message to the console

If 0 Operator option – retry or cancel.

If 1 Operator option – ignore, retry or cancel. Return to the user.

ON value for the third operand of the CCB macro: X'0800'.

Bit 6 – Return on:

- DASD read or read-verify data check.
- PRT1 printer passback requested (see also Note 1 on page 61).
- Tape read data check.
- Punch equipment check on an IBM 2520 or 2540.
- Permanent error on an IBM 3505 or 3525.
- Equipment check on an IBM 3881.
- IBM 3895 error codes

If 0 Operator option:

- Ignore or cancel for tapes and for card punches.
- Retry or cancel for DASD.

If 1 For an error on a PRT1 printer (see also Note 1 on page 61), tape, or DASD, return to the user after physical IOCS attempted to correct the error.

For a permanent error on an IBM 3505 or 3525, bit 3 of byte 3 is also set to 1.

For an IBM 3895, data checks on count are not retained. Error codes are returned in CCB byte 8; refer to the IBM 3895 Document Reader/Inscriber manuals for information about these codes.

ON value for the third operand of the CCB macro: X'0200'.

Bit 7 – User error routine (see also Note 2 on page 61):

If 0 A physical IOCS error routine is used, except when the CCB senseaddress operand is specified. The latter requires error recovery by the user program.

If 1 User handles error recovery. You cannot handle channel-control and interface-control checks. When a channel-data, unit, or channel-chaining check occurs, the system sets on bit 2 of byte 2 and completes posting and dequeuing. Incorrect length and unit exception are treated as normal conditions (posted with completion). You must also request device-end posting (bit 5 of byte 2) to obtain error information after channel end.

ON value for the third operand of the CCB macro: X'0100'.

CCB Byte 3

Bit 0 – Check bit:
If 1 Indicates one of the following:
- Data check in DASD count field.
- For an IBM 33xx CKD disk – permanent I/O error.
- For a PRT1 printer (see also Note 1 on page 61) – a print (equipment) check.
- For an IBM 3540 – A special record has been transferred. A deleted or bad spot record was read. After the read-in of the special record, the CCW chain is broken.

Bit 1 – See "If 1" below:
If 1 Indicates one of the following:
- DASD track overrun.
- For a PRT1 printer (see also Note 1 on page 61) – a print quality error (equipment check).

Bit 2 – See "If 1" below:
If 1 Indicates one of the following:
- End of DASD cylinder.
- For a PRT1 printer (see also Note 1 on page 61) – a line-position error. A line-position error can occur as a result of an equipment, data, or FCB-parity check.

Bit 3 – See "If 1" below:
If 1 Indicates one of the following:
- Tape read data check (see "Note" below).
- For an IBM 2540, or 3881 – equipment check (see "Note" below).
- Disk data check (see "Note" below).

  Note: Operation was unsuccessful. Bit 2 of byte 2 is set to 1; bit 0 of byte 3 is set to 0.

- For an IBM 3203, – one of the following types of equipment checks: print, print-data, print-clutch, and read (see "Note" below).
- For an IBM 3505 or 3525 – permanent I/O error (see "Note" below).
- For a PRT1 printer (see also Note 1 on page 61) – a data/print check.
- For an IBM 3540 – data check.

  Note: Bit 6 of byte 2 is set to 1.

Bit 4 – Nonrecovery questionable condition:
If 1 Indicates one of the following:
- For card I/O – unusual command sequence.
- For a DASD – no record found.
- For a PRT1 printer (see also Note 1 on page 61) – UCB parity check (command retry).

Bit 5 – Operator option:

Applies to a retry on DASD (see also Note 2 on page 61).
If 0 The nonrecovery questionable condition bit is set to 1, and control is passed to the requesting program
If 1 Retry operation is performed for the no-record-found condition. System will initiate appropriate action if error persists after a limited number of retries (bit 3 in byte 2).

ON value for the third operand of the CCB macro: X'0004'.

Bit 6 – See "If 1" below and also Note 2):

If 1 Indicates one of the following:

- Verify error for DASD.
- Carriage channel 9 overflow (on a printer), but only if bit 5 of byte 2 is set to 1.

Bit 7 – Command-chain retry:

Specify the bit to be set to 1 if you use command chaining (see also Note 2).

If 0 Retry begins at the first CCW (or channel program).

If 1 Retry begins at the last CCW that was executed.

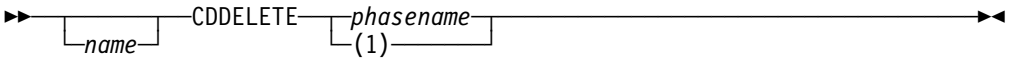
- If an error occurs, physical IOCS updates the CCW address in bytes 9 through 11 of the CCB. Your program therefore must restore the original CCW address before it starts another I/O operation using the same CCB.

ON value for the third operand of the CCB macro: X'0001'.

Notes:

1. Applies also to a 4248 operating in native mode, except where this device is excluded explicitly.
2. User-option bits; set in CCB macro. The system sets the other bits off while processing the EXCP macro; it sets them on if the specified condition occurs.

CDDELETE (Delete Loaded Phase) Macro



Requirements for the caller:

AMODE: 24 or 31

RMODE: 24 or ANY

ASC Mode: Primary

The macro deletes a phase previously loaded by a CDLOAD request. Deletion means that the phase load count is decremented by one. If the load count is zero, the phase entry in the CDLOAD directory (the anchor table) will be cleared and the GETVIS storage occupied by the phase is freed.

CDDELETE returns the actual load count of a phase in register 0.

If the maximum load count was exceeded by previous CDLOAD requests, a CDDELETE request is ignored and return code 4 is given.

phasename | (1)

For phasename, specify the name of the phase to be deleted. If register notation is used, the register must contain the address of an 8-byte field that holds the phase name as an alphameric character string.

The address of phasename is regarded as a 24-bit or 31-bit address, depending on the AMODE (24 or 31) of the caller.

Return Codes in Register 15

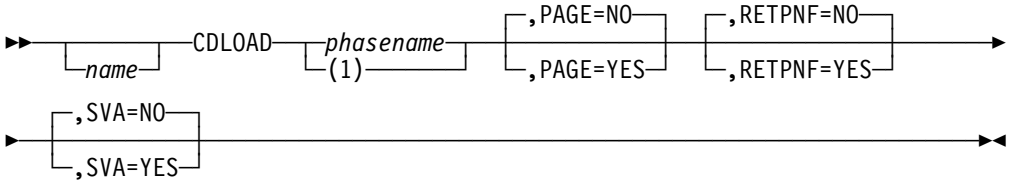
After execution of the macro, register 15 contains one of the following return codes:

0 CDDELETE completed successfully.

4 CDDELETE was given for a phase whose maximum load count was exceeded. The phase is not deleted.

20 Phase not found. This return code is also given for phases that are loaded into the SVA, because SVA phases cannot be deleted by CDDELETE.

CDLOAD (Control-Directory Load) Macro



Requirements for the caller:

AMODE: 24 or 31

RMODE: 24 or ANY

ASC Mode: Primary

The macro causes the phase specified as the first operand to be loaded from a sublibrary into the partition GETVIS area. The phase is loaded into virtual storage either below 16MB or anywhere, as indicated by the phase's RMODE. The phase is loaded only if it is not yet in either the partition GETVIS area or the SVA. CDLOAD returns control to the phase which issued the macro.

The CDLOAD macro must not be used for a phase that has been linked as a member of an overlay structure. Instead, use the LOAD macro without specifying a load address.

phasename | (1)
For phasename, specify the name of the required phase. If register notation is used, the register must contain the address of an 8-byte field that holds the phase name as an alphameric character string.

The address of phasename is regarded either as a 24-bit or 31-bit address, depending on the AMODE (24 or 31) of the caller.

PAGE=NO | YES
If you want to have the phase loaded on a page boundary, specify PAGE=YES.

RETPNF=NO | YES
Determines whether the issuing phase is canceled if the phase to be loaded does not exist in a sublibrary. With RETPNF=YES, the phase is not canceled; instead, control is returned to the issuing phase with the appropriate return code.

SVA=NO | YES
SVA=YES specifies that CDLOAD is to provide the loadpoint/entrypoint of SVA phases **without** loading them.

When a phase is to be loaded, CDLOAD:

1. Determines the size of the phase
2. Acquires the required amount of GETVIS storage
3. Loads the phase into that storage.
4. CDLOAD maintains a load count for each loaded phase. This load count is incremented by one when a CDLOAD request for the phase is given, and it is decremented by one for a CDELETE request. If the load count reaches the

maximum (65535), it is not increased by any following CDLOAD request, but left at the maximum. Any further CDDELETE request will be rejected (with return code 4).

After a phase has been loaded or if a phase need not be loaded (because it is already in the partition GETVIS area or in the SVA), the output registers contain the following values:

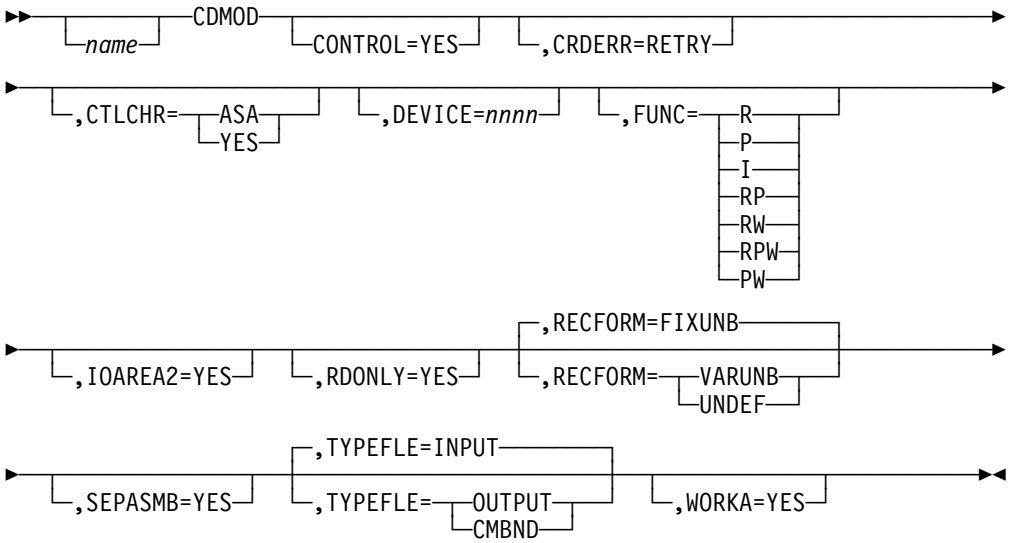
0 The load address
1 The entry point address. CDLOAD sets the high-order bit in register 1 to indicate the phase's AMODE, which it obtains from the phase's directory information: either to 0 (for AMODE 24) or to 1 (for AMODE 31). If the phase's AMODE is ANY, CDLOAD sets the high-order bit in register 1 corresponding to the caller's AMODE.
14 The length of the phase
15 The return code

Return Codes in Register 15

After execution of the macro, register 15 contains one of the following return codes:

0 CDLOAD completed successfully.
4 The size of the (real) partition's GETVIS area is 0K.
8 The specified length exceeds the GETVIS area.
12 Insufficient storage available in the GETVIS area.
16 The partition CDLOAD directory (also known as anchor table) is full and there is no space (system GETVIS area) available to allocate a new anchor table.
20 The phase does not exist in a sublibrary (this return code occurs only with RETPNF=YES).
24 A move-mode phase was requested.

CDMOD (Card I/O Module Definition) Macro



Requirements for the caller:

AMODE: 24

RMODE: 24

ASC Mode: Primary

The CDMOD macro defines a logic module for a card reader/punch file. It is also to be used for the IBM 3881 optical reader. If you do not provide a name for the module, IOCS generates a standard module name.

CONTROL=YES
Include this operand if the CNTRL macro is used with the module and its associated DTFs. The module also processes files for which the CNTRL macro is not used.

If this operand is specified, the CTLCHR operand must not be specified.

This operand cannot be specified if IOAREA2 is used for an input file or if an input file is used in association with a punch file (when the operand FUNC=RP or RPW is specified) on the IBM 3525; in this case, however, this operand can be specified in the DTFCD and CDMOD for the associated punch file.

CRDERR=RETRY
Include this operand if error retry routines for the IBM 2540 and 2520 punch-equipment check are included in the module. Whenever this operand is specified, any DTF used with the module must also specify the same operand. This operand does not apply to an input or a combined file.

CTLCHR=ASA | YES
Include this operand if first-character stacker select control is used. ASA denotes the American National Standards character set, YES the System/370* character set (see Appendix A, “Control Character Codes” on page 429). Any DTF to be used with this module must have the same

operand. If CTLCHR is included, CONTROL must not be specified. This operand does not apply to a combined file or to an input file.

DEVICE=nnnn
For nnnn, specify one of the following IBM device codes:

2540
2520
3505
3525
3881

Include this operand to specify the I/O device used by the module.

FUNC=R | P | I | RP | RW | RPW | PW
This operand specifies the type of file to be processed by the IBM 3525. Any DTF used with the module must have the same operand. R indicates read, P indicates punch, and W indicates print.

When FUNC=I is specified, the file will be both punched and interpreted; no associated file is necessary to achieve this.

RP, RW, RPW, and PW specify associated files; when one of these operands is specified for one file, it must also be specified for the associated file(s). Associated files can have only one I/O area each.

IOAREA2=YES
Include this operand if a second I/O area is used. Any DTF used with the module must also include the IOAREA2 operand. This operand is not required for combined files. This operand is not valid for associated files.

RDONLY=YES
This operand causes a read-only module to be generated. Whenever this operand is specified, any DTF used with the module must have the same operand.

RECFORM=FIXUNB | VARUNB | UNDEF
This operand specifies the record format: fixed-length, variable-length, or undefined. Any DTF used with the module must have the same operand. If TYPEFLE=INPUT, TYPEFLE=CMBND, or FUNC=I, RECFORM must be FIXUNB. For the IBM 3881, only RECFORM=FIXUNB is valid, which is also the default.

SEPASMB=YES
Include this operand only if the module is to be assembled separately. This produces an object module ready to be cataloged into a suitable sublibrary either by the standard name or by the user-specified name. The module name is used as the module's transfer address. If you omit this operand, the assembler assumes that the module is assembled together with the DTF in your program.

TYPEFLE=INPUT | OUTPUT | CMBND
This operand generates a module for either an input, output, or combined file. Any DTF used with the module must have the same operand. For the IBM 3881, only TYPEFLE=INPUT is valid, which is also the default.

WORKA=YES

This operand must be included if records are to be processed in work areas instead of in I/O areas. Any DTF used with the module must have the same operand. This operand is not valid for the IBM 3881.

Standard CDMOD Names

Each name begins with a 3-character prefix (IJC) and continues with a 5-character field corresponding to the options permitted in the generation of the module.

CDMOD name = IJCabcde

Char.	Content	Specified Option
a	F	RECFORM=FIXUNB (always for INPUT, CMBND, or FUNC=I files)
	U	RECFORM=UNDEF
	V	RECFORM=VARUNB
b	A	CTLCHR=ASA (not specified if CMBND)
	C	CONTROL=YES
	Y	CTLCHR=YES
	Z	CTLCHR or CONTROL not specified
c	B	RDONLY=YES and TYPEFLE=CMBND
	C	TYPEFLE=CMBND
	H	RDONLY=YES and TYPEFLE=INPUT
	I	TYPEFLE=INPUT
	N	RDONLY=YES and TYPEFLE=OUTPUT
	O	TYPEFLE=OUTPUT
d	B	WORKA=YES and IOAREA2
	I	IOAREA2=YES
	W	WORKA=YES
	Z	WORKA and IOAREA2 not specified
	Z	WORKA=YES not specified (CMBND file only)
e	0	DEVICE=2540
	2	DEVICE=2520
	4	DEVICE=2540 and CRDERR
	5	DEVICE=2520 and CRDERR
	6	DEVICE=3505
	7	DEVICE=3525 and FUNC=R/P or omitted
	A	DEVICE=3525 and FUNC=RP
	B	DEVICE=3525 and FUNC=RW
	C	DEVICE=3525 and FUNC=PW
	D	DEVICE=3525 and FUNC=I
	E	DEVICE=3525 and FUNC=RPW
	P	DEVICE=3881

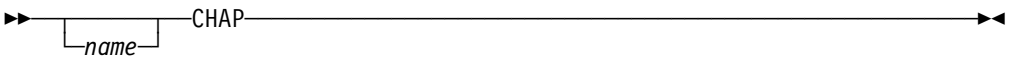
Subset/Superset CDMOD Names

All but one of the operands are exclusive (that is, do not allow supersetting). A module name specifying C (CONTROL) in the **b** location is a superset of a module name specifying Z (no CONTROL or CTLCHR). A module with the name IJCFCIW0 is a superset of a module with the name IJCFZIW0.

```
          *  *  *  *  *
I  J  C   F  A  B  B  0
          V  Y  C  I  2
          U  +  H  W  4
             C  I  Z  5
             Z  N     6
                O     7
                      A
                      B
                      C
                     ...
                      M
                      N
                      O
                      P
```

+ Subsetting/supersetting permitted.
***** No subsetting/supersetting permitted.

CHAP (Change Priority) Macro



Requirements for the caller:

AMODE: 24 or 31

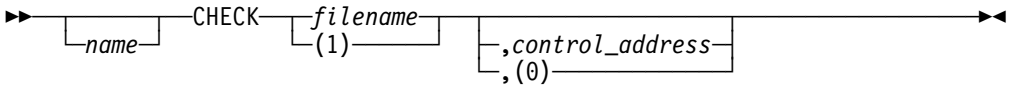
RMODE: 24 or ANY

ASC Mode: Primary

The macro lowers the priority of the issuing subtask. The issuing subtask now becomes the subtask with the lowest priority of all the subtasks within the partition.

A CHAP macro issued by the main task is ignored.

CHECK (Check I/O Completion) Macro



Requirements for the caller:

AMODE: 24

RMODE: 24

ASC Mode: Primary

The macro prevents processing until data transfer on an I/O operation is complete. It may be issued either after a READ or WRITE macro was issued for a work file, or after a READ was issued for a MICR file to ensure that data transfer is complete.

Because of differences in the way that IOCS posts CCB transmission information bits in the DTFs, you should always issue a CHECK macro to ensure that data transfer is complete before testing these bits. If the data transfer is completed without an error or other exceptional condition, CHECK returns control to the next sequential instruction. If an error condition is encountered, control is transferred to the ERROPT address. If ERROPT is not specified, processing continues at the next instruction. If end-of-file is encountered, control transfers to the EOFADDR address.

Issuing a CHECK macro after a READ on a MICR device allows you to query the MICR document buffer (see Figure 3 on page 70) and to specify the control_address operand.

filename | (1)
The operand specifies the name of the file associated with the record to be checked or, if register notation is used, the register containing a pointer to the field that contains this name. This name is the same as that specified for the DTFxx header entry for the file.

control_address | (0)
Indicates the address to which control passes when a buffer is waiting for data or when the file is closed. If register notation is used, the specified register must point to a field that contains this address.

The CHECK macro determines whether the MICR document buffer:

- Contains data ready for processing.

  In this case, control passes to the next sequential instruction.

- Is waiting for data.

  In this case, or if the **file is closed**, control passes to the address specified for control-address, if present. Else, control passes to the address specified in the ERROPT operand of the DTFMR macro.

- Contains a special non-data status.

In this case, control passes to the ERROPT routine. In the routine, you can examine the posted error conditions before determining your action (see byte 0 of the document buffer, bits 2, 3, and 4).

The CHECK macro also determines whether the file (filename) is closed.

Return from the ERROPT routine to the next sequential instruction by a branch either on register 14 or to the address in register 0.

If an error, a closed file, or a waiting condition occurs with neither control-address nor an ERROPT address specified, control is given to your program at the next sequential instruction.

If the waiting condition occurred, byte 0, bit 5 of the buffer is set to 1. If the file was closed, byte 0, bits 5 and 6 of the buffer are set to 1.

Figure 3 (Page 1 of 2). MICR Document Buffer Format

Byte	Bit	Comment
0		Buffer Status Indicator
	0	The document is ready for processing (you need not test this bit).
	1	Irrecoverable stacker select error, but all document data is present. You may continue to issue GETs and READs.
	2	Irrecoverable I/O error. An operator I/O error message is issued. The file is unusable and must be closed.
	3	Unit Exception. You requested disengage and all follow-up documents are processed. The LITE macro may be issued, and the next GET or READ engages the device for continued reading.
	4	Intervention required or disengage failure. This buffer contains no data. The next GET or READ continues normal processing. This indicator allows your program to give operator information needed to select pockets for documents not properly selected and to determine unread documents.
	5	The program issued a READ, no document is ready for processing; bits 0 through 2 of byte 0 are off, or the file is closed (bit 6 of byte 0 is on). The CHECK macro examines this bit. Test bits 1 through 4 and take appropriate action. No data from a buffer should be processed if bits 2, 3, or 4 are on.
	6	The program issued a GET or READ and the file is closed. Bit 5 also on.
	7	Reserved.
1		Buffer Status Indicator
	0	Applies to old devices.
	1-7	Reserved.
2		Buffer Status Indicator
	0	Applies to old devices.
	1-3	Reserved.

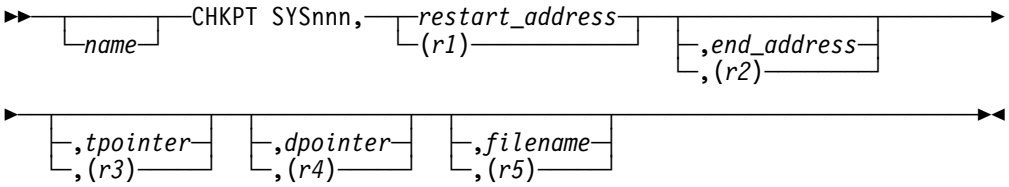
Figure 3 (Page 2 of 2). MICR Document Buffer Format

Byte	Bit	Comment
		Bits 4 through 7 reflect MICR sense information
	4	Data check occurred while reading. Examine byte 3 to determine the error fields.
	5	Overrun occurred while reading. Examine byte 3 to determine the error fields. Overruns cause short-length data fields.
	6-7	The meanings of these bits depend on the device type, the model, and the engineering-change level of (old) MICR devices.
3		Buffer Status Indicator – The byte contains MICR sense information (applies to old devices).

4		Buffer Status Indicator – Pocket code
		
		Should be examined by your stacker select routine. If bits 2, 3, or 4 of byte 0 are on, this byte is X'00'. No document was read and your stacker selection routine was not entered.
		
		If auto-selection occurs, this value is ignored. A no-op (X'03') is issued to the device, and the reject pocket code (X'CF') is placed into byte 5. The possible pocket codes (when bit 6 or 7 of byte 2 is on) are:
		
		`Pocket 0: X'0F'` `Pocket 7: X'7F'`
		`Pocket 1: X'1F'` `Pocket 8: X'8F'`
		`Pocket 2: X'2F'` `Pocket 9: X'9F'`
		`Pocket 3: X'3F'` `Reject Pocket: X'CF'`
		`Pocket 4: X'4F'`
		`Pocket 5: X'5F'`
		`Pocket 6: X'6F'`

5		Buffer Status Indicator – Pocket-selected code Indicates the pocket selected for the document. The contents of the byte are normally the same as that in byte 4. X'CF' is inserted whenever auto-selection occurs (bit 6 of byte 2, bit 7 of byte 2, bit 0 of byte 2, or bit 2 of byte 3). These conditions may result from late READ commands, errant document spacing, or late stacker selection: • Start I/O for stacker selection is unsuccessful (bit 1 of byte 0).
6–end		Additional User Work Area This area can be used as a work or an output area or both. The size of this area is determined by the DTFMR ADDAREA operand. The only size restriction: this area, plus the status-indicator bytes and the data portion must not exceed 256 bytes. This area may be omitted. Document Data Area This area follows immediately your work area. In this area, the data is right-adjusted. The length of this area is determined by the DTFMR RECSIZE operand.

CHKPT (Checkpoint Request) Macro



Requirements for the caller:

AMODE: 24

RMODE: 24

ASC Mode: Primary

The CHKPT macro causes the status of your program to be recorded in one of two ways:

- As interspersed records of a tape output file.
- Into a separate file on disk.

Should your program come to an abnormal end, then you can restart the program by submitting job control statements as required. The partition in which the program is to be restarted must begin at the same location as when the program was checkpointed. Also, its end address must not be lower than the end address when the checkpoint was taken. For more information about restarting a checkpointed program, see the *VSE/ESA Guide to System Functions*under “Starting a Program from a Checkpoint”.

If the CHKPT macro is processed successfully, your program receives control with register 0 containing, in unpacked decimal format, the number of the checkpoint. If the macro is processed unsuccessfully, that is, no checkpoint has been taken, register 0 contains zero. In addition, the reason for the failure is printed on SYSLOG.

Note: If a program using routines in the SVA is being checkpointed, you must make sure that SVA routines occupy the same locations on restart, should a restart become necessary.

Special register notation cannot be used with any of the CHKPT macro operands.

All VSAM files must be closed before the CHKPT macro is issued. A SAM ESDS (supported by the VSE/VSAM Space Management for SAM feature) cannot be repositioned by the restart program.

Restrictions:

Checkpoint/restart does not support the 31-bit environment; the macro is canceled when issued from a partition that crosses the 16MB line.

Checkpoint/restart does not support data spaces; that is, data spaces that a program may access are not recorded during CHKPT requests.

Checkpoint/restart does not support dynamic partitions.

Checkpoint/restart does not support PFIXed pages which are PFIXed in page frames outside the ALLOCR area; this means, whenever the CHKPT macro is used, any currently PFIXed page must have been PFIXed in the ALLOCR area. The easiest way to ensure this is to have no PFIX limit set (via the JCL SETPFIX statement) during the execution of your program. The CHKPT macro is ignored when it detects a page being PFIXed in a page frame outside the ALLOCR area.

The IBM 9346 tape is not allowed as a checkpoint device. If a program requests a checkpoint to be written to a 9346 tape, the function issues a message and ignores the request.

SYSnnn
Specifies the logical unit on which the checkpoint information is to be stored. It must be an EBCDIC magnetic tape or a DASD volume.

restart_address | (r1)
Specifies a symbolic name of the instruction (or register containing the address) at which execution is to restart if processing must be continued later.

end_address | (r2)
A symbolic name assigned to (or register containing the address of) the uppermost byte of the program area required for restart. This address must be higher than the highest address of storage occupied by any phase loaded into the partition. The address should be a multiple of 2K. If the address is not a multiple of 2K, it is rounded to the next 2K boundary. If this operand is omitted, all storage allocated to the partition (other than the GETVIS area) is checkpointed.

The specified end address is ignored if any GETVIS request was executed in the partition. (Note that GETVIS storage may have been requested by included IBM routines). In this case again, all storage allocated to the partition is checkpointed.

tpointer | (r3)
Address of an 8-byte field containing 2 V-type address constants used in repositioning magnetic tape at restart time. The address may be a symbolic address or contained in a register.

The first constant points to a table containing the file names of all logical IOCS magnetic tape files to be repositioned. Each file name points to the corresponding DTF table where IOCS maintains repositioning information.

The second constant points to a table containing information for physical IOCS magnetic tape files to be repositioned. The entries in the table are:

- First halfword: hexadecimal representation of the logical unit address of the tape (copy from CCB).
- Second halfword: number of files within the tape (in binary notation), that is, the number of tape marks between the beginning of the tape and the position at checkpoint.
- Third halfword: number (in binary notation) of physical records between the preceding tape mark and the position at checkpoint.

If the first, second, or both constants are zero, no tapes are repositioned.

If the tables are contained in the same source module as the CHKPT macro, the constants must be defined as A-type constants.

Both tables have to be preceded by a halfword containing the number of table entries.

dpointer | (r4)
Address of a DASD operator verification table, used to allow the operator to verify DASD volume serial numbers at restart time. May be a symbolic address or contained in a register.

The entries in the table must consist of the following two halfwords:

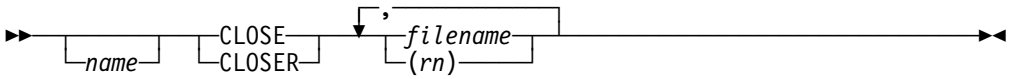
- The logical unit number (in hexadecimal notation) of each DASD unit used by your program (copied from CCB bytes 6 and 7).
- Reserved.

The table has to be preceded by a halfword containing the number of table entries.

There must be one entry for each DASD unit to be verified. At restart time, the volume serial number of each of these DASD units is printed on SYSLOG.

filename | (r5)
This operand applies only if checkpoint records are to be written into a separate file on disk. The operand specifies the name of the associated DTFPH.

CLOSE and CLOSER (Close a File) Macro



Requirements for the caller:

AMODE: 24 or 31

RMODE: 24

ASC Mode: Primary

The format of the CLOSER macro is the same as that of the CLOSE macro, except that you code CLOSER instead of CLOSE in the operation field.

When CLOSER is specified, the symbolic address constants that CLOSER generates from the parameter list are self-relocating. When CLOSE is specified, the symbolic address constants are not self-relocating. Throughout the manual the term CLOSE also implies CLOSER, unless stated otherwise.

The CLOSE or CLOSER macro is used to deactivate previously opened files; it ends the association between a logical file declared in a program and a specific physical file on an I/O device. Issuing a CLOSE or CLOSER macro for a file ensures that the system properly ends processing for the specified file(s).

A file may generally be closed at any time, with the following exceptions:

- Console files must not be closed; the CLOSE(R) macro is invalid for files defined by means of the DTFCN.
- A file may not be closed from within an ERROPT routine.

If issued for a 4248 printer file making use of the horizontal copy function, the macro causes horizontal printing to be turned off.

If CLOSE or CLOSER is issued to an unopened tape input file, the option specified in the DTF rewind option is performed. If CLOSE or CLOSER is issued to an unopened tape output file, no tapemark or labels are written.

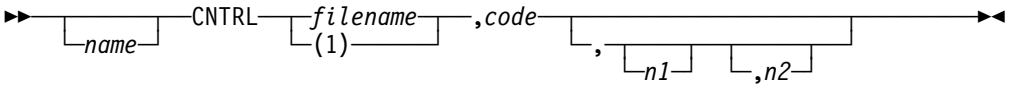
No further requests can be issued for the closed file until it is reopened.

filename | (rn)

Code the symbolic name of the file (DTF filename) to be closed. You can close up to 16 files with one macro by coding additional file names. Alternatively, you can load the address of a file name into a register and specify the register, using ordinary register notation.

The high-order 8 bits of this register must be zeros. For CLOSER, the address of the file's name may be pre-loaded into any of the registers 2 through 12. For CLOSE, this address may be pre-loaded into register 0 or any of the registers 2 through 12.

CNTRL (Control Device) Macro



Requirements for the caller:

AMODE: 24

RMODE: 24

ASC Mode: Primary

The CNTRL (control) macro provides commands for I/O devices. These commands apply to non-data operations of an I/O unit and are specific to this unit. They specify functions such as rewinding tape, card stacker selection, and line spacing on a printer. For optical readers, commands specify marking error lines, correcting a line for journal tapes, document stacker selecting, or ejecting and incrementing documents.

The CNTRL macro does not wait for completion of the command before it returns control to your program, except when certain codes are specified for optical readers.

If your program issues CNTRL requests, ensure that your DTFxx macros (except DTFMT and DTFDR):

- Include the CONTROL operand.
- Omit the CTLCHR operand.

If your program uses control characters although the CONTROL operand is specified, then IOCS does not recognize them and treats them as data.

Conversely, if your DTFxx macro for a file includes the CTLCHR operand, your program normally cannot issue a CNTRL macro for this DTFxx. It can issue a CNTRL macro in this case only if:

1. The macro refers to a DTFPR defined printer and requests an immediate printer operation, and
2. The device being used is a PRT1 printer or an IBM 4248.

Examples of immediate printer operations are: space or skip immediate, enable or disable horizontal copying.

The CNTRL macro is ignored if specified for DTFSD or DTFDI DASD files. The macro cannot be used for:

- A card-input file that is being processed with two I/O areas.
- A card-input file on an IBM 3525 if this file is associated with a punch file. Use the macro for the associated punch file instead.

filename | (1)
Must be the name of the file specified in the DTF header entry. It can be specified as a symbol or in register notation.

code Is the mnemonic code for the command to be performed. A list of the possible command codes and the related n1 and n2 values is given following the description of the operands.

n1 Is required whenever a number is needed for stacker selection, immediate printer carriage control, or for line or page marking on the IBM 3886.

n2 Applies to delayed spacing or skipping or to timing mark check on the IBM 3886. In the case of a printer file, the operands n1 and n2 may be required. If n1 is omitted and n2 is specified, a comma must be coded for n1.

A list of valid command codes and related n1 and n2 values follows. The codes are listed by device class (disk, tape, and so on) or, if necessary, by device type.

CKD-Disk Devices

Code	n1	n2	Requested Operation
SEEK			Seek to address.

Magnetic Tape Units

Code	n1	n2	Requested Operation
REW			Rewind tape.
RUN			Rewind and unload tape.
ERG			Erase gap (writes blank tape).
WTM			Write tapemark.
BSR			Backspace to inter-record gap.
BSF			Backspace to tapemark.
BSL			Backspace to logical record. See also "Note" under FSL below.
FSR			Forward-space to inter-record gap.
FSF			Forward-space to tapemark.
FSL			Forward-space to logical record.
SYN			Synchronize buffer and program. Applies only if the tape unit is a buffered device. Causes all data held in the buffer to be written to tape.

Note: The codes BSL and FSL can be used to control the processing of spanned records. LIOCS ignores a CNTRL request with BSL or FSL if a RELSE macro immediately precedes the CNTRL request.

Printers – Any Type

Code	n1	n2	Requested Operation
SP	c[1]	d[2]	Carriage-space c or d lines. For c or d, you give the number of lines to be spaced. This number can be 1, 2, or 3. See also [3].

Code	n1	n2	Requested Operation
SK	c[1]	d[2]	Skip to channel c or d or both. See also [3] and [4].
UCS			Applies to IBM printers with a UCS buffer, either standard or as a feature.
	ON		Data checks are processed with an indication to the operator.
	OFF		Data checks are ignored, and blanks are printed.

1 A value for n1 requests immediate printer control (before printing).

2 A value for n2 indicates delayed printer control (after printing).

3 Applies to IBM printers of type 1403, 3203, 3800, PRT1 (including 4248 in native mode), and 3525 card punch with the print feature.

4 If an IBM 3525, a skip to channel 1 is valid only for a print-only file.

PRT1 Printers – Including IBM 4248 in Native Mode

Note: If the printer does not support the requested operation, the system ignores your request.

Code	n1	n2	Requested Operation
FOLD			Print uppercase characters for any byte with equivalent bits 2 through 7.
UNFOLD			Print character equivalents for any EBCDIC byte.
CLRPRT			Clear print buffer. Applies if the printer has a data buffer. Causes all data contained in the buffer to be printed before the program receives control again.
ORDER			Applies only to an IBM 4248 operating in native mode.
	DHC		Disable horizontal copy. Stops horizontal copy printing.
	EHC		Enable horizontal copy. Starts horizontal copy printing if your printer's FCB contains an FCB image with a horizontal copy control code. Horizontal copy printing is turned off automatically when one of the following occurs: • Your program issues a CLOSE (or CLOSER) macro for the printer file. • The system initiates an automatic close for the printer file at the end of the job step.
	PURDAT		Purge data. Causes all data stored in the printer's data buffer to be erased.

Card I/O Devices

Code	n1	n2	Requested Operation
SS	1 or 2		Select stacker 1 or stacker 2. Applies to IBM 2520.
PS	1 to 3		Select stacker 1, 2, or 3. Applies to IBM 2540, 3505, and 3525. For a 3505 or 3525, the value 3 defaults to stacker 2.

IBM 3881 Optical Mark Reader

Code	n1	n2	Requested Operation
PS	1 or 2		Select stacker 1 or stacker 2.

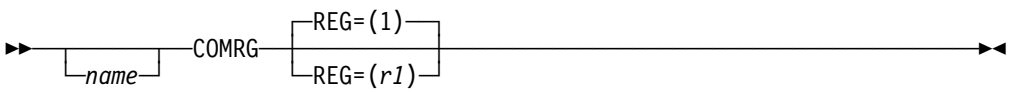
IBM 3886 Optical Character Reader

Code	n1	n2	Requested Operation
INC			Increment document at read station.
DMK	see [1]		Page-mark the document when it is stacker-selected as specified by n1.
LMK	see [2]		Line-mark the document when it is stacker-selected as specified by n1.
ESP	1 or 2	see [1]	Eject and stacker-select the current document to stacker A or B. Perform line-mark station timing-mark check as specified by n2.

[1] name(r) number

[2] name(r) number,number

COMRG (Communication Region Access) Macro



Requirements for the caller:

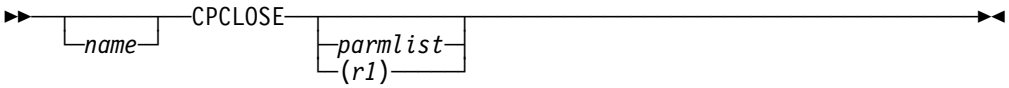
AMODE: 24 or 31

RMODE: 24 or ANY

ASC Mode: Primary

The COMRG macro places the address of the communication region of the partition from which the macro is issued into the specified register. If the operand is omitted, register 1 is assumed.

CPCLOSE (Control Program File Close) Macro



Requirements for the caller:

AMODE: 24

RMODE: 24

ASC Mode: Primary

The macro can be used in spooling programs written in Basic Assembler Language. It causes a CPCLOSE command to be issued to VM/ESA* in order to release a print or punch file for output.

parmlist | (r1)

This operand specifies a 16-byte parameter list. The format of the list is described below. In your program, you must set up the list before you issue the macro. If the parameter list name is specified, the system loads the address into register 1. If a register is specified, it is assumed to contain the address of the parameter list and this address is loaded into register 1. If no operand is specified, register 1 is assumed to contain the address of the parameter list. The format of this list is:

Bytes	Contents
0-1	Always 0.
2-3	Device address in hexadecimal format.
4-7	Device address in EBCDIC format.
8-15	Job name (left justified).

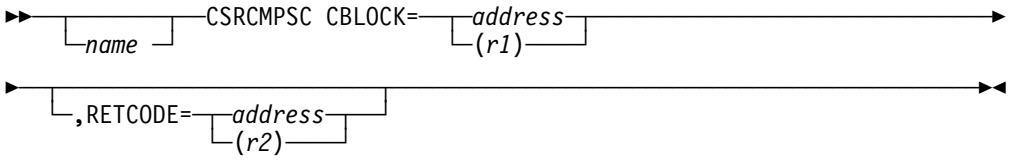
Assume the device at address 280 is to be closed and the name of the job is JOB. The parameter list then contains:

```
00 00 02 80  F0 F2 F8 F0  D1 D6 C2 40  40 40 40 40
```

Return Codes in Register 15

00 Successful completion of CPCLOSE macro.
04 Device is invalid, no CLOSE is issued.
08 Supervisor not running under VM/SP.

CSRCMPSC (Compression/Expansion) Macro



Requirements for the caller:

AMODE: 31

ASC Mode: Primary or AR (access register)

The CSRCMPSC macro is used to compress data (to save DASD space, for example) and to expand previously compressed data. Data compression and expansion services allow you to compress certain types of data so that the data occupies less space while you are not using it. You can then restore the data to its original state when you need it.

For detailed guide information on how to compress and expand data, see "Compressing and Expanding Data" in the manual *VSE/ESA System Macro User's Guide*.

CBLOCK=address | (r1)
Specifies the address of a 36-byte input/output area (compression block) which contains the parameter information for the compression/expansion service. The area is mapped by DSECT CMPSC in macro CSRYCMPS and contains information such as the compression and expansion dictionaries and the source and target areas, together with their lengths.

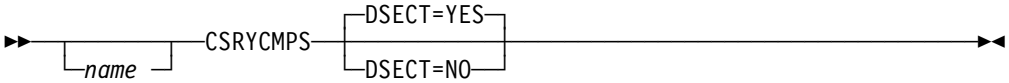
RETCODE=address | (r2)
Specifies the address of an 8-byte area into which the return code is to be copied from R15.

Return Codes

00 Successful completion of CSRCMPSC macro.
04 Source operand was not completely processed. No room is left in the source operand.
10 A field in the CSRYCMPS area does not contain a value.
14 The symbol size (CMPSC_SYMSIZE) in the CSRYCMPS area does not have a value of 1 through 5.
18 The target area for compression or the source area for expansion is not large enough to hold even one compression symbol. The length of the area is specified in the CSRYCMPS area.
1C The length of the string represented by a single compression symbol exceeds the limit of 260 bytes.
20 The number of child characters for a compression dictionary entry exceeds 260.
24 A compression dictionary entry indicates that it contains more than six child characters, not including sibling characters.
28 The number of extension characters for a compression dictionary entry with 0 or 1 child characters exceeds 4.

2C A sibling descriptor compression dictionary entry has a count of 0.
30 Expansion of a compression symbol used more than 127 dictionary entries.

CSRYCMPS (Map Compression Control Block) Macro



Required RMODE: **24**

The CSRYCMPS macro causes a DSECT of the compression control block to be generated.

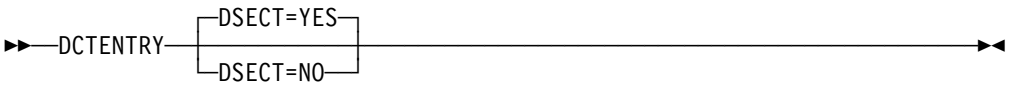
DSECT=YES | NO
YES causes a mapping DSECT to be generated. NO causes inline code to be generated.

CSRYCMPS

CSRYCMPS is described in the MVS/ESA Compression manual.

Input from Sinram!

DCTENTRY (Map GETVCE) Macro

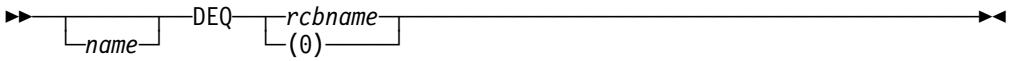


Required RMODE: **24**

The DCTENTRY macro describes device characteristics retrieved with the GETVCE macro.

DSECT=YES | NO
YES causes a mapping DSECT to be generated. NO causes inline code to be generated.

DEQ (Dequeue Resource) Macro



Requirements for the caller:

AMODE: 24 or 31

RMODE: 24 or ANY

ASC Mode: Primary

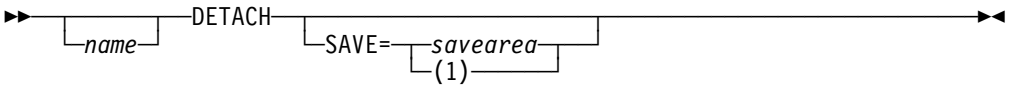
A task releases a resource by issuing the DEQ macro. If other tasks are enqueued on the same RCB, the DEQ macro frees from their wait condition all other tasks that were waiting for that resource. A task that attempts to dequeue a resource that was not enqueued or that was enqueued by another task is abnormally terminated. Dequeuing under these two conditions within an abnormal termination routine results in a null operation instruction.

The DEQ macro supports the 31 bit environment. DEQ may be issued in 24-bit or 31-bit addressing mode, above or below the 16MB line.

When DEQ is issued in AMODE 24, the RCB address is treated as a 24-bit address. When DEQ is issued in AMODE 31, the RCB address is treated as a 31-bit address.

rcbname | (0)
The operand is the same as that in the ENQ macro and specifies the address of the RCB.

DETACH (Detach Task) Macro



Requirements for the caller:

AMODE: 24 or 31

RMODE: 24 or ANY

ASC Mode: Primary

The macro terminates execution of a task. A subtask is normally terminated by issuing a DETACH macro in the main task, attaching subtask, or in the subtask itself.

The macro sets byte 2, bit 0 of the ECB to 1 (if specified in the ATTACH macro) to indicate task termination. All tasks waiting on this ECB are taken out of the wait state.

If the **subtask** issues a DETACH macro without an operand, only the subtask issuing the DETACH macro is terminated. Any subtasks attached by the terminating subtask are not affected by the termination.

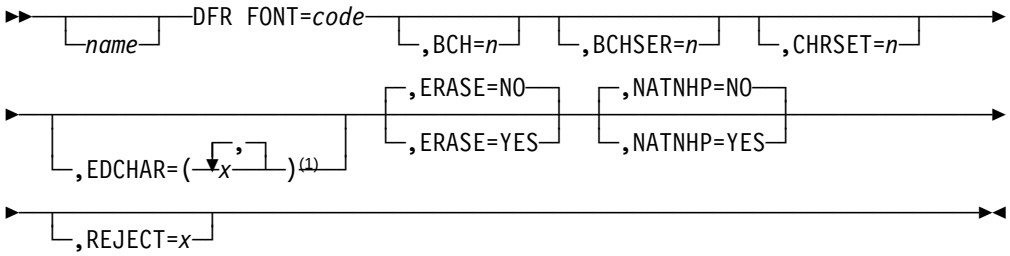
If the **main task** issues the DETACH macro without specifying an operand, it will be canceled, that is, all processing in the partition is terminated abnormally. However, the partition may recover, dependent on the AB exit specification.

SAVE=savearea | (1)
A subtask may also terminate a subtask which it attached by issuing the DETACH macro with the SAVE operand. If the main task issues the DETACH macro with the SAVE operand, it can terminate any subtask in the partition. The SAVE operand provides the address of the save area specified in the ATTACH macro for the subtask to be terminated. The save area has to be allocated below the 16MB line.

When the task identified by SAVE cannot be detached by the owning task, the request is ignored. When the address specified by SAVE does not point to the save area of an active task, the result is unpredictable.

Note: If the subtask being terminated uses VSAM files, ensure that these files are closed before you issue this macro.

DFR (Define Font Record) Macro



Note:
[1] Can be repeated up to six times.

Required RMODE: **24**

The DFR macro defines attributes common to a group of line types on an IBM 3886.

FONT=code
The operand is required. It specifies the default font for all fields described by the format record. The default font is used to read a field unless another font is specified for an individual field through the DLINT macro. The valid codes and the fonts they represent are:

ANA1 = Alphameric OCR-A font (mode 1)
ANA2 = Alphameric OCR-A font (mode 2)
ANB1 = Alphameric OCR-B font
GOTH = Gothic font
MRKA = Mark OCR-A font
MRKB = Mark OCR-B font
NHP1 = Numeric hand printing (normal mode)
NHP2 = Numeric hand printing (verify mode)
NUMA = Numeric OCR-A font
NUMB = Numeric OCR-B font (mode 3)

For a description of these fonts, see the appropriate IBM 3886 device manuals.

BCH=1 | 2 | 3
This operand is valid only if the serial numbering feature is installed. The operand indicates that batch numbering is to be performed. Specifying 1, 2, or 3 indicates that documents routed to a stacker are to be batch numbered. Specifying 1 indicates stacker A, 2 indicates stacker B, 3 indicates both stackers. If this operand is specified, the BCHSER operand is invalid. For more information on batch numbering, see the IBM 3886 device manuals.

BCHSER=1 | 2 | 3
This operand is valid only if the serial numbering feature is installed. The operand indicates that both batch and serial numbering are to be performed. Specifying 1, 2, or 3 indicates that documents routed to a stacker are to be batch and serial numbered. Specifying 1 indicates stacker A, 2 indicates stacker B, 3 indicates both stackers. If this operand is specified, the BCH operand is invalid. If neither BCH nor BCHSER is specified, batch and serial numbering are not performed. For more

information on batch and serial numbering, see the IBM 3886 device manuals.

CHRSET=0 | 1 | 2 | 3 | 4 | 5

Specifies which one of the options in Figure 4 is to be used for recognizing characters. If this operand is not supplied, 0 is assumed.

Figure 4. Character Set Option List

OCR-A			OCR-B			
Numeric Mode	**Alphameric Modes**		**Numeric Mode**	**Alphameric Mode**		
High-Speed Printers or Typewriters	**Mode 1 (High-Speed Printer)**	**Mode 2 (Typewriter)**	**High-Speed Printers or Typewriters**		**Hexa-decimal Code**	**Format Record Codes**
$	$	$	$	$	5B	00
£	£	£	£	£	5B	01
¥	¥	¥	¥	¥	5B	02
	Ñ	Ñ		Ñ	7B	
$	$	$	$	$	5B	03
	Å	Å		Å	5B	
	Æ	Æ		Æ	7B	
	Ø	Ø		Ø	7C	04
	$	$		Ü Note	5B	
	Ä	Ä		Ä	7B	
	Ö	Ö		Ö	7C	
$	Ü	Ü			F0	05

Note: In OCR-A font the Ü is coded as a zero and should be used only in alphabetic fields.

EDCHAR=(x,...)

Specifies up to six characters that may be deleted from any field that is read. The operand EDITn=EDCHAR of the DLINT macro controls this function for individual fields. If this operand is omitted, no character deletion is performed. See the note under the REJECT operand discussion for characters that must be specified in quotes. For example, to specify the characters &, >, and), you would code EDCHAR=('&','>',')').

ERASE=NO | YES

Specifies whether group and character erase symbols are to be recognized as valid symbols. If this operand is not specified, NO is assumed. For more information on group and character erase symbols, see the IBM 3886 device manuals.

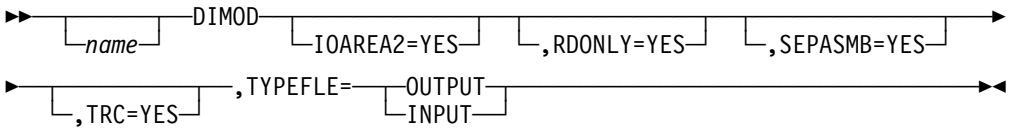
NATNHP=NO | YES

Specifies which of the numeric hand printing character set options are used for the numbers 1 and 7. YES indicates that the European Numeric Hand Printing (ENHP) characters 1 and 7 are used; NO indicates the Numeric Hand Printing (NHP) characters 1 and 7 are used. If this operand is not coded, NO is assumed.

REJECT=x
Indicates the character that is to be substituted in the data record for any reject character read by the device. If this operand is omitted, X'3F' is assumed. Reject characters are characters that are not recognizable by the device.

Note: This note applies to the keywords REJECT and EDCHAR. Apostrophes enclosing the character are optional for all characters except special characters used in macro operands. For a description of these characters, see the *Assembler Language* manual.

DIMOD (Device-Independent I/O Module Definition) Macro



Requirements for the caller:

AMODE: 24

RMODE: 24

ASC Mode: Primary

The DIMOD macro defines a logic module for a device-independent file. If you do not provide a name for the module, IOCS generates a standard module name.

For DASD devices and for PRT1 and 3800 printers, a user-supplied logic module is not required. If one is supplied, it is ignored. OPEN always provides linkage to an IBM-supplied logic module which resides in the SVA.

IOAREA2=YES
Include this operand if a second I/O area is needed. A module with this operand can be used with DTFDI specifying either one or two I/O areas. If the operand is omitted or is invalid, one I/O area is assumed.

RDONLY=YES
This operand causes a read-only module to be generated. Whenever this operand is specified, any DTF used with the module must have the same operand.

SEPASMB=YES
Include this operand only if the module is to be assembled separately. This produces an object module ready to be cataloged into a suitable sublibrary either by the standard name or by the user-specified name. The module name is used as the module's transfer address. If you omit this operand, the assembler assumes that the module is assembled together with the DTF in your program.

TRC=YES
Include this operand to specify whether the module is to test the table reference character indicator in the DTFDI or ignore that indicator. If TRC=YES is specified, the generated module can process output files with

table reference characters and those without. If the TRC operand is specified, TYPEFLE=INPUT must not be specified.

TYPEFLE=OUTPUT | INPUT
Include this operand to specify whether the module is to process input or output files. If OUTPUT is specified, the generated module can process both input and output files.

Standard DIMOD Names

Each name begins with a 3-character prefix (IJJ) followed by a 5-character field corresponding to the options permitted in the generation of the module.

DIMOD name = IJJabcde

Char.	Content	Specified Option
a	F	Always
b	C	RPS=SVA is not specified
	V	RPS=SVA
c	B	TYPEFLE=OUTPUT (both input and output)
	I	TYPEFLE=INPUT
d	I	IOAREA2=YES
	Z	IOAREA2=YES is not specified
e	C	RDONLY=YES
	D	RDONLY=YES is not specified

Subset/Superset DIMOD Names

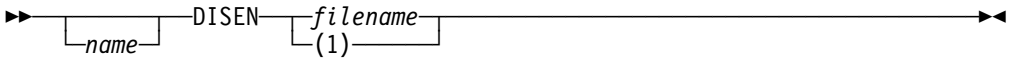
All of the operands except TRC=YES allow subsetting. A module name specifying B is a superset of the module specifying I, for example. IJJFCBID is a superset of the module IJJFCIID.

The IBM-supplied preassembled logic modules do not have TRC=YES. The system programmer can reassemble them with TRC=YES to support 3800 table reference characters. Although the code that is generated for a module assembled with TRC=YES is different from the code that is generated for a module with TRC=NO, the module name is the same. If some, but not all DIMOD logic modules are reassembled this way, it may interfere with subsetting or supersetting.

```
                *  +  +  *
   I  J  J  F  C  B  I  C
                V  I  Z  D
```

+ Subsetting/supersetting permitted.
***** No subsetting/supersetting permitted.

DISEN (Disengage Document Reader) Macro



Requirements for the caller:

AMODE: 24

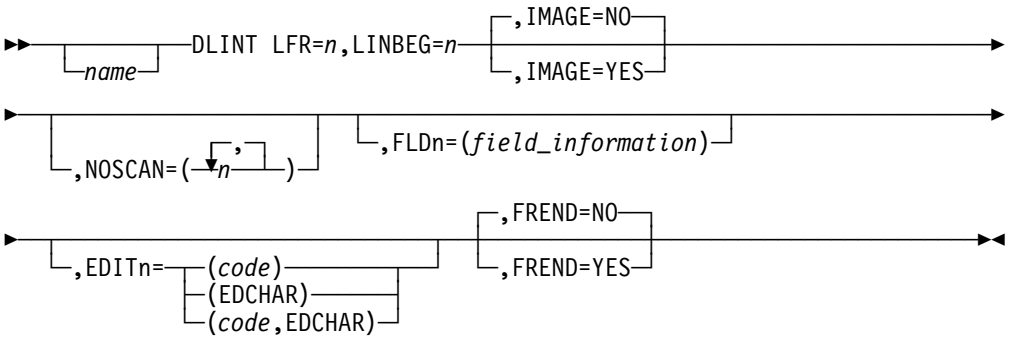
RMODE: 24

ASC Mode: Primary

This macro stops the feeding of documents through the magnetic character reader or optical reader/sorter. The program proceeds to the next sequential instruction without waiting for the disengagement to complete. You should continue to issue GET or READ until the unit exception bit (byte 0, bit 3), of the buffer status indicators is set on (see Figure 3 on page 70).

filename | (1)
Specifies the name of the file to be disengaged. This name is the same as that specified for the DTFMR header entry for the file. The operand can be specified either as a symbol or in register notation.

DLINT (Define Line Type) Macro



Required RMODE: **24**

The macro describes one line type in a format group and the individual fields in the line.

Line-Information Specifications

LFR=n

This operand specifies the line format record number for the line. The decimal number specified must be in the range of 0 through 63. The line format record describes the format of one type of line; the line format record number is used to identify the line format record. This number is specified in the READ macro when you read a line of data from a document.

LINBEG=n

This operand specifies the beginning of a line. The beginning position is the distance, measured in units of 0.1 inch (2.54 mm), from the left edge of the document to the left boundary of the first field. The limiting range of this position is 4 to 85.

IMAGE=NO | YES

This operand specifies whether the data record should be in standard mode (IMAGE=NO), or image mode (IMAGE=YES). If this operand is not specified, IMAGE=NO is assumed.

NOSCAN=(n,...)

Specifies an area on the document line that is to be ignored by the IBM 3886. 'n' is a decimal number indicating the distance, measured in units of 0.1 inch (2.54 mm), from the left edge of the document to the right end of the NOSCAN field. The field immediately to the left of the NOSCAN field must end with an address delimiter rather than a character delimiter.

Field-Information Specifications

FLDn=(field_information)

The operand describes each of the fields of a line.

n of FLDn

Is a number from 1 up to 14. The following rules apply to the use of the keyword FLDn:

- Fields may be described in any order in the macro.

- Each EDITn operand must follow its associated FLDn operand.
- The n suffix need not be 1 for the first field in the line; however, the n suffix must increase for each field from left to right on the document line.

m An **address delimiter**, which you code as a decimal number. It specifies the distance, measured in units of 0.1 inch (2.54 mm), from the left edge of the document to the right end of the field being defined. The last field in a line must end with an address delimiter.

x A **character delimiter**. It specifies the character that indicates the end of a field. The character delimiter is not considered part of the data; it is neither included in the data record nor used in determining the length of the field.

Apostrophes enclosing the characters are optional for all characters except 0 through 9 and the special characters used in macro operands. For these characters, the apostrophes are required. For a description of these characters, see the *Assembler Language* manual.

If a field ends with a character delimiter, the next field must be read using a font from the same font group. The font groups are:

- NPH1, NPH2, GOTH
- ANA1, ANA2, NUMA, MRKA
- NUMB, MRKB
- ANB1

length
Is a decimal number that specifies the length of the field in the edited record. Specify a value from 1 to 127. If IMAGE=NO is specified, this specification is required; if IMAGE=YES is specified, this specification is invalid.

The length you specify refers to the length of the field after any EDITn options have been performed. The sum of the field lengths for a line cannot be greater than 130.

NCRIT
Indicates that this is not a critical field. If this specification is omitted, the field is assumed to be critical.

fontcode
Specifies a font for this field, different from the font specified in the DFR macro. If this specification is omitted, the font specified in the DFR macro is used for the field. For information about the valid codes, see the description of the FONT=code operand of the DFR macro on page 85.

EDITn=(code) | (EDCHAR) | (code,EDCHAR)
Describes the editing functions to be performed on the data by the IBM 3886.

The specifications are the same for keywords EDIT1 through EDIT14. There must be an FLDn keyword corresponding with each EDITn keyword you specify. If an EDITn keyword is specified, a code, EDCHAR, or both must be specified. When image mode is used, an EDITn keyword is invalid.

When the editing functions are completed and the field is greater than the specified length, the field is truncated from the right and the wrong length field indicator is set on in the header record. If only blanks are truncated, the wrong length field indicator is not set.

code specifies the blanks to be removed and the fill characters to be added to the field, if any. The valid codes and their meanings are:

ALBHIF
All blanks are removed, the data is right-justified and the field is padded with EBCDIC zeros (X'F0') on the left.

ALBLOF
All blanks are removed from the data, the data is left-justified and the field is padded with blanks on the right. If code is omitted, ALBLOF is assumed.

ALBNOF
All blanks are removed; the data must be equal in length to the field length specified. No padding is done.

HLBHIF
All high- and low-order blanks are removed, the data is right-justified and the field is padded to the left with EBCDIC zeros (X'F0') (see Note, below).

HLBLOF
All high- and low-order blanks are removed, the data is left justified, and the field is padded with blanks on the right (see Note, below).

NOBLOF
No blanks are removed, the data is left-justified, and the field is padded on the right with blanks.

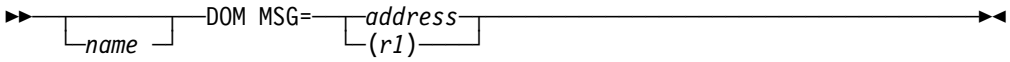
Note: Two consecutive embedded blanks is the maximum number sent.

EDCHAR indicates that the characters specified in the EDCHAR keyword of the DFR macro are to be deleted from the field. If this specification is omitted, the characters are not deleted.

If the EDITn keyword is omitted or if EDITn=EDCHAR is specified and the code is omitted, ALBLOF is assumed.

FREND=NO | YES
Indicates whether this is the last DLINT macro for the format record. NO indicates that more DLINT macros follow; YES indicates that this is the last one. If this operand is omitted, NO is assumed.

DOM (Delete Operator Message) Macro



Requirements for the caller:

AMODE: 24 or 31

RMODE: 24

ASC Mode: Primary

The DOM macro may be used to delete a console message, when the condition that caused the message to be issued does not exist any more (for example, a device became ready).

The message being deleted may never appear on any console. If it did appear, any highlighting or hold state is removed as a result of DOM.

MSG=address | (r1)
Specifies the address of a 4-byte area containing the ID of the message to be deleted. Registers 1 to 12 may be used for register notation. The message ID was returned by the WTO or WTOR macro used to issue the message.

The only effect of the DOM macro is to remove the message from the 'hold' state (reply or action pending) and to reset its intensity attribute from 'high' to 'normal', as applicable. The message is still routed and logged as if no DOM macro had been issued.

DOM for a message written via WTO may be issued by a different task than the one that issued the original message identified by the message ID. In particular, it may be issued by a console application, rather than by the application that originated the message, when the message is being deleted as a result of an operator request.

DOM for a message written with WTOR may only be issued by the task that issued the original message; otherwise it is ignord.

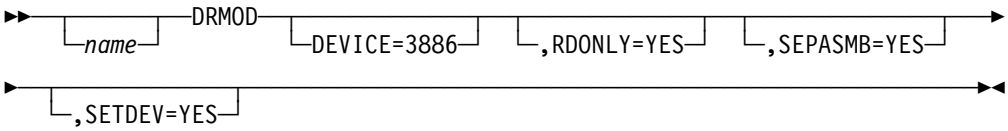
Return Codes in Register 15

0 Successful completion.

Cancel Codes

21 One or more input parameters are invalid or not supported by VSE/ESA.
25 One or more of the specified addresses are invalid.
45 Mode violation (for example, caller is in AR-mode).

DRMOD (Document Read Module Definition) Macro



Requirements for the caller:

AMODE: 24

RMODE: 24

ASC Mode: Primary

The macro defines a logic module for a file on an IBM 3886. If you do not provide a name for the module, IOCS generates a standard module name.

DEVICE=3886
Specifies that the IBM 3886 is the input device. This operand may be omitted.

RDONLY=YES
This operand generates a read only module. RDONLY=YES must also be specified in the DTF. For additional programming requirements concerning this operand, see the DTFDR RDONLY operand.

SEPASMB=YES
Must be specified if the I/O module is to be assembled separately. This entry produces an object module ready to be cataloged into a suitable sublibrary either by the standard name or by the user-specified name. The module name is used as the module's transfer address. If you omit this operand, the assembler assumes that the module is assembled together with the DTF in your program.

SETDEV=YES
Is specified if the SETDEV macro may be used when processing a file with this I/O module. If SETDEV=YES is specified in the DRMOD macro but not in the DTFDR macro, the SETDEV macro cannot be used when processing that file.

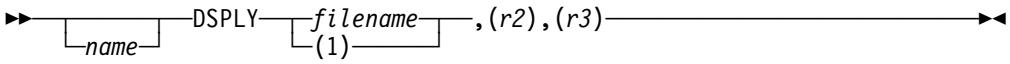
Standard DRMOD Names

Each name consists of eight characters: IJMZxxD0. The fifth and sixth characters are variables as follows:

- If SETDEV=YES is specified, the fifth character is S; otherwise it is Z.
- If RDONLY=YES is specified, the sixth character is R; otherwise it is Z.

Note: Subsetting/supersetting is allowed with the SETDEV keyword, but not with the RDONLY keyword.

DSPLY (Display Document Field) Macro



Requirements for the caller:

AMODE: 24

RMODE: 24

ASC Mode: Primary

The DSPLY macro displays the document field on the display scope of your IBM 1287. A complete field may be keyboard-entered if a 1287 read error makes this type of correction necessary. An unreadable character may be replaced by the reject character either by the operator (if processing in the online correction mode) or by the device (if processing in the offline correction mode). You may then use the DSPLY macro to display the field in error.

filename | (1)
Is the symbolic name specified in the name field of the DTFOR macro for your 1287 file.

(r2) Specifies any of registers 2 to 12. This register must contain the address of the load format CCW that gives the document coordinates for the field to be displayed. When the DSPLY macro is used in the COREXIT routine, the address of the load format CCW can be obtained by subtracting 8 from the 3-byte address that is right-justified in the fullword location beginning at filename+32 (the high-order fourth byte of this fullword should be ignored). If the DSPLY macro is not used in the COREXIT routine, you must determine the load format CCW address.

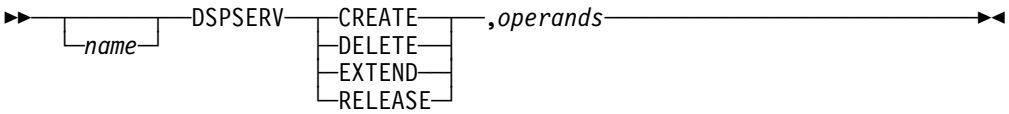
(r3) Specifies any of registers 2 to 12. This register must contain the address of the load format CCW that gives the coordinates of the reference mark associated with the displayed field.

DSPSERV (Data Space) Macro

The DSPSERV macro creates, deletes, and controls data spaces. A data space is a range of up to two gigabytes of contiguous virtual storage addresses that a program can directly manipulate through ESA/370* instructions. Unlike an address space, a data space can hold only data and programs stored as data. For detailed guide information on how to create and use data spaces, see “Chapter 8, Creating and Using Data Spaces” in the manual *VSE/ESA Extended Addressability*.

The DSPSERV macro can only be executed in 31-bit addressing mode.

The DSPSERV macro supports the following main functions:



CREATE
Create a data space

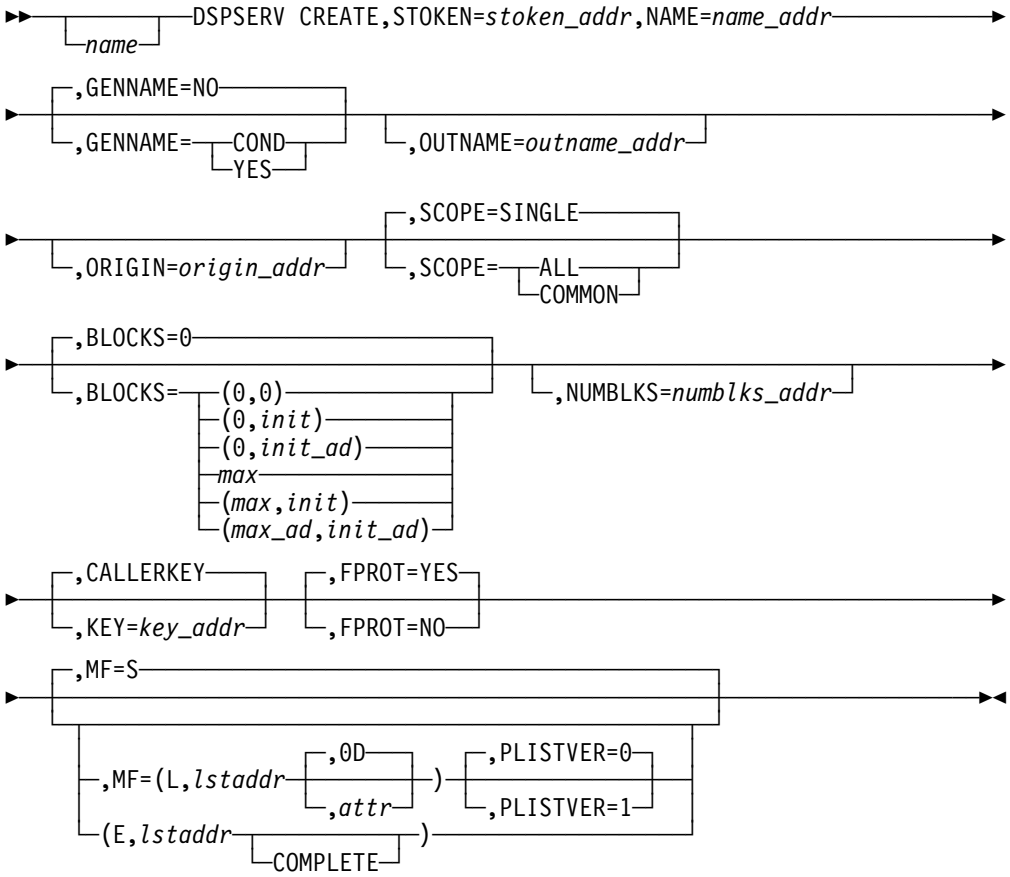
DELETE
Delete a data space

EXTEND
Increase the current size of a data space

RELEASE
Release a data space

For a detailed description of the main functions, see the “DSPSERV (Data Space) Macro” with the corresponding keyword (DSPSERV CREATE, DSPSERV DELETE,...).

DSPSERV CREATE (Create Data Space) Macro



Requirements for the caller:

AMODE: 31

RMODE: 24 or ANY

ASC Mode: Primary or AR (access register) if SYSSTATE ASCENV=AR

DSPSERV CREATE requests the system to create a data space. Creating a data space can be compared to issuing a GETVIS request for storage. VSE gives you contiguous 31-bit virtual storage of the size you specify and initializes the storage to hexadecimal zeros. The entire data space has the storage key that you request or, by default, the storage key that matches your own PSW key.

STOKEN=stoken-addr

Specifies the address of an eight-byte field where the system returns the STOKEN (space token or identifier) that uniquely identifies the data space.

Your program can then gain access to the data space by using the ALESERV ADD macro to add an entry to an access list and obtain an ALET (access list entry token) for the given STOKEN. The entry on the access list identifies the newly created data space, and the ALET indexes the entry.

NAME=name-addr

Specifies the address of an eight-byte variable or constant that contains the name of the data space. The naming conventions are described under “Data Space Naming Conventions” on page 101.

GENNAME=NO | COND | YES

Specifies whether or not you want the system to generate a name for the data space to ensure that all names are unique within the partition. The system generates a name by adding a 5-character prefix (consisting of a numeral followed by four characters) to the first three characters of the name you supply in the NAME operand. For example, if you supply XYZDATA in the NAME operand, the name becomes nCCCCXYZ where 'n' is the numeral, CCCC is the 4-character string generated by the system, and XYZ comes from the name you supplied on NAME. See “Data Space Naming Conventions” on page 101 for more information.

GENNAME=NO The system does not generate a name. In the NAME operand you must supply a name that is unique within the partition. GENNAME=NO is the default.

GENNAME=COND The system generates a unique name only if you supply a name that is already being used. Otherwise, the system uses the name you supply.

GENNAME=YES The system takes the name you supply in the NAME operand and makes it unique.

If you want the system to return the unique name it generates, use the OUTNAME operand.

OUTNAME=outname-addr

Specifies the address of an eight-byte variable where the system returns the data space name it generated if you specified GENNAME=YES or GENNAME=COND.

ORIGIN=origin-addr

Specifies the address of a four-byte variable where the system returns the beginning address (either zero or 4096) of the data space. The system tries to start all data spaces at origin zero; on some processors, however, the origin is 4096. To be independent of the type of processor, IBM recommends always to use ORIGIN.

SCOPE=SINGLE | ALL | COMMON

SCOPE=SINGLE indicates that the data space may be accessed only by programs running in the owner's partition. The ALESERV ADD macro (which the program must issue to gain access to the data space) adds an entry for the data space only to access lists of the partition where the program owning the data space is running.

SCOPE=ALL indicates that the data space can be accessed by programs running in the owner's partition (SCOPE=SINGLE) **and** in other selected partitions. This allows to share data selectively among programs running in different partitions. Whenever a program running in another partition wants to access the data space (for example, called DX) it must first set up an ALESERV ADD request with that STOKEN (DX) in order to get an ALET returned. Thus, access to the data space can be restricted by communicating the STOKEN only to certain programs.

SCOPE=COMMON indicates that the data space can be used by **all** programs and from all partitions in the system. Such a data space provides a commonly addressable area similar to the Shared Virtual Area (SVA).

To gain access to the data space, a program must issue the ALESERV ADD macro with AL=PASN. ALESERV ADD then adds an entry for the data space to the caller's PASN-AL and returns the ALET for that entry. Additionally, ALESERV ADD adds the same entry to the access lists (PASN-ALs) of **all** active partitions in the system. Also, newly created partitions receive the same entry. All programs running in the system use the same ALET to access the data space. Since the entry is now on all PASN-ALs, programs in other partitions do not have to issue the ALESERV ADD macro. The creating program must only pass the ALET for the data space to the other programs.

Any program can create and delete SCOPE=SINGLE data spaces. However, only PSW key 0 programs can create, extend, release, or delete SCOPE=ALL or SCOPE=COMMON data spaces.

For a summary of the rules for creating, deleting, and using data spaces with different SCOPEs, see “Chapter 8, Creating and Using Data Spaces” in the manual *VSE/ESA Extended Addressability*.

BLOCKS=0 | (0,0) | (0,init) | (0,init-ad) | max | (max,init) | (max-ad,init-ad)

Specifies the **maximum** and the **initial** size of the data space or the size of an area within the data space, as shown in Figure 5.

- The first (or only) value is always the maximum size, which identifies the largest amount of storage you will need in the data space.
- The second value which you specify (after the comma) denotes an initial size which identifies the amount of storage you will immediately use.

BLOCKS=	Maximum Size	Initial Size
0	default	default
(0,0)	default	default
(0,init)	default	init
max	max	default
(max,init)	max	init

Example 1: If you specify BLOCKS=(0,500), the system sets the maximum size of your data space to the default value and the initial size to 500 blocks.
Example 2: If you specify BLOCKS=1000, the system sets the maximum size to 1000 and the initial size to the default value.

Figure 5. Maximum and Initial BLOCKS Specification

BLOCKS=0 (which is the default) or **BLOCKS=(0,0)** establishes a data space with the maximum size **and** the initial size both set to the default size, which is either the IBM defined default or the default set via the SYSDEF DSPACE,DFSIZE job control (or attention) command. The system returns this default (as the maximum size) at the location identified by NUMBLKS.

BLOCKS=(0,...) sets the maximum size to the default value.

BLOCKS=(...,init) specifies the number of 4K-blocks to be used as the initial size of the data space to be created. This initial size is the amount of storage you will immediately use. (If you need more space in the data space, you can use the DSPSERV EXTEND macro to increase the available storage.) If the initial size you specify exceeds or equals the maximum size, the initial size becomes the maximum size.

Note: The amount of storage taken from VSIZE is always the initial number of 4K-blocks you specify, rounded up to the next multiple of 8, if necessary. For example, if you specify 10 blocks, the system rounds this number up to 16 and takes 64K from VSIZE.

BLOCKS=(...,init-ad) specifies the address of a field that contains the initial size of the data space.

BLOCKS=max or **BLOCKS=(max,...)** specifies the maximum size (in number of blocks) to which the data space can be expanded. A block is a unit of 4 K bytes. You cannot extend the data space beyond its maximum size. The maximum size that can be specified is 524,288 blocks (the product of 524,288 times 4K bytes being 2 Gigabytes). Note, however, that your installation can set limits to the amount of storage available for all data spaces together; for details, see the SYSDEF job control (or attention) command.

BLOCKS=(max-ad,...) specifies the address of a field that contains the maximum size of the data space to be created.

NUMBLKS=numblks-addr
Specifies the address of a four-byte area where the system returns the maximum size (in blocks) of the newly created data space.

If you specify BLOCKS=0 or omit the BLOCKS operand when creating a data space, the system returns the default.

CALLERKEY
Indicates that the data space has the storage key that matches the PSW key of the caller.

KEY=key-addr
Specifies the address of an eight-bit variable or constant that contains the storage key of the data space to be created. The key must be in bits 0-3 of the field. The system ignores bits 4-7.

Note: A program running with a non-zero PSW key can only delete a data space it owns; the PSW key has to match the storage key of the data space.

FPROT=YES | NO
Specifies whether or not the data should be fetch-protected. Fetch protection means that a program must be in the key of the data space storage (or key 0) to reference data in the data space. If you specify YES, the entire data space is fetch-protected.

MF=S Specifies the **standard** form of the macro, which is used to place the parameters into an inline parameter list. This is also the default if the MF parameter is omitted.

MF=(L,...

Specifies the **list** form of the macro, which is used to construct a non-executable control program parameter list.

lstaddr specifies the address of the area that the system is to use for the parameter list.

attr is an optional 1- to 60-character input string which can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code **attr**, the system provides a value of 0D, which forces the parameter list to a double-word boundary.

PLISTVER=0|1 specifies which parameter list the system is to generate. 0 produces a 56-character parameter list; 1 (which is always recommended) produces a 60-character list.

No other parameters may be specified if the list form of the macro is chosen.

MF=(E,...

Specifies the **execute** form of the macro, which uses the parameter list generated by the list form of the macro.

lstaddr specifies the address of the parameter list. This address must not be in a data space. If the caller of the macro is in 24-bit mode, the address of the parameter list must not be above the 16MB line.

COMPLETE specifies that the system is to check for required parameters and supply optional parameters that are not specified.

Return Codes in Register 15 (and Reason Codes in Register 0)

00 Successful completion
04 Successful completion. 2 Gigabytes (524,288 blocks) were requested. However, since the processor does not support a data space with zero origin, a data space with one block less than specified (524,287 blocks) was created.
08 R0=X'xx0005xx': Creation of data space would violate installation criteria. The available storage may be exhausted. The installation criteria is defined with SYSDEF DSPACE... and can be displayed with QUERY DSPACE.
08 R0=X'xx0009xx': GENNAME=NO was specified, but the data space name is not unique within the partition.
08 R0=X'xx0012xx': The system's set of generated names for data spaces has been temporarily exhausted.
0C The system cannot create any additional data spaces at this time because of a shortage of resources:
0C R0=X'xx000600': No system GETVIS storage available (page manager).
0C R0=X'xx000601': No virtual storage available (page manager).
0C R0=X'xx000602': No real storage available (page manager).

Data Space Naming Conventions

Data space names are from one to eight bytes long. They can contain letters, numbers, and the characters @, #, and $, but no imbedded blanks. Names that contain fewer than eight bytes must be left-justified and padded on the right with blanks.

Data space names must be unique within the partition of the data space owner. No other data space belonging to the partition can have the same name. Therefore, in choosing names for your data spaces, avoid using the same names that IBM products use for data spaces. IBM products use the following names for data spaces:

- Names that begin with A through I, where the first three characters denote an IBM component prefix, if possible.
- Names that begin with SYSAxxxx through SYSIxxxx, where the fourth through sixth characters denote any IBM component prefix, if possible.
- Names that begin with numbers or the characters SYSDS or SYSIV.

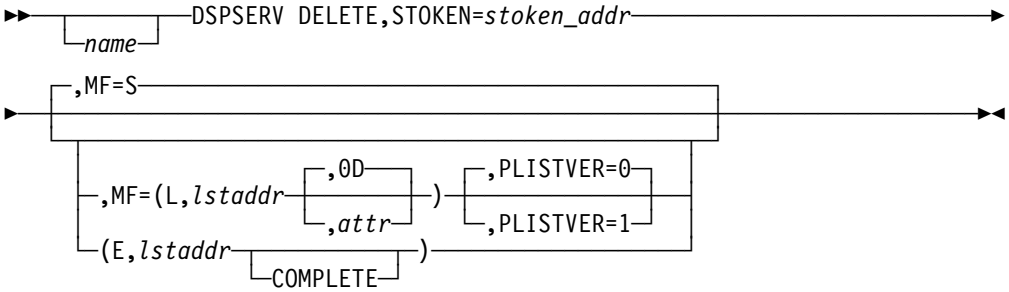
Use The Following Names For Your Data Spaces

- **Problem state programs (non-zero PSW key)** can use data space names that begin with @, #, $, or the letters J through Z, with the exception of SYS. The system abends problem state programs that begin any names with SYS (check for subsystem).
- **Supervisor state programs, programs with PSW key 0, and subsystems** can use data space names that begin with @, #, $, or letters I through Z. In addition, they can use names that begin with SYSJ through SYSZ. The system abends programs whose names begin with SYSDS.

  Use names that begin with SYSJ through SYSZ to ensure that the names of the data spaces belonging to supervisor state programs and programs with PSW key 0 do not conflict with the names of data spaces that belong to problem state programs.

When you choose a name, consider that operators have to identify data space names in some display requests and the DUMP command.

DSPSERV DELETE (Delete Data Space) Macro



Requirements for the caller:

AMODE: 31

RMODE: 24 or ANY

ASC Mode: Primary or AR (access register) if SYSSTATE ASCENV=AR

DSPSERV DELETE requests the system to delete a data space (when your program does not need it any longer). Before you delete the data space, you should remove the data space entry (ALET) from the access list by means of the ALESERV DELETE macro.

A non-zero key program can delete any data space it owns, provided its PSW key matches the storage key of the data space. A key-0 program can delete any data space it owns and other data spaces of the caller's partition.

STOKEN=stoken-addr
Specifies the address of the eight-byte STOKEN for the data space. (returned from DSPSERV CREATE).

MF=S Specifies the **standard** form of the macro, which is used to place the parameters into an inline parameter list. This is also the default if the MF parameter is omitted.

MF=(L,...
Specifies the **list** form of the macro, which is used to construct a non-executable control program parameter list.

lstaddr specifies the address of the area that the system is to use for the parameter list.

attr is an optional 1- to 60-character input string which can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code **attr**, the system provides a value of 0D, which forces the parameter list to a double-word boundary.

PLISTVER=0|1 specifies which parameter list the system is to generate. 0 produces a 56-character parameter list; 1 (which is always recommended) produces a 60-character list.

No other parameters may be specified if the list form of the macro is chosen.

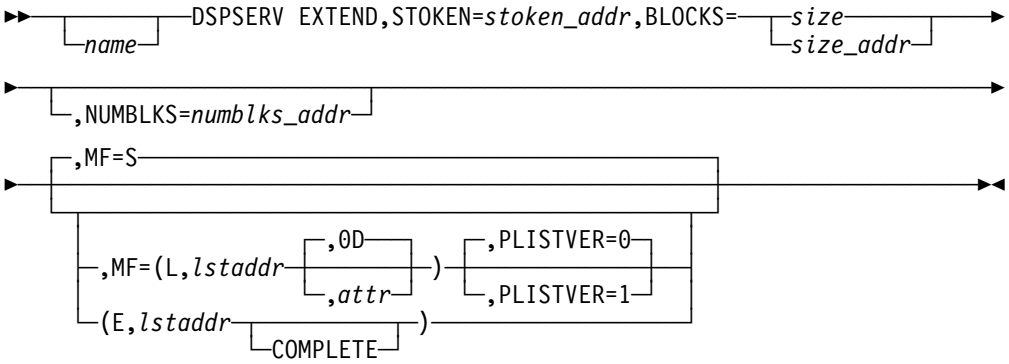
MF=(E,...
Specifies the **execute** form of the macro, which uses the parameter list generated by the list form of the macro.

lstaddr specifies the address of the parameter list. This address must not be in a data space. If the caller of the macro is in 24-bit mode, the address of the parameter list must not be above the 16MB line.

COMPLETE specifies that the system is to check for required parameters and supply optional parameters that are not specified.

The return code in register 15 is always 0.

DSPSERV EXTEND (Extend Data Space) Macro



Requirements for the caller:

AMODE: 31

RMODE: 24 or ANY

ASC Mode: Primary or AR (access register) if SYSSTATE ASCENV=AR

DSPSERV EXTEND requests the system to increase the current size of the data space. Use the EXTEND function only for a data space that was created with an initial size smaller than the maximum size.

Before you can reference storage beyond the current size, you must use EXTEND to increase the storage that is currently available. If you reference data space storage beyond the current size, the system rejects the request and terminates the caller with a cancel code.

A caller with non-zero PSW key can extend the data space it owns. A key-0 program can extend any data space it owns and other data spaces of the caller's partition.

The system rejects the request if the extended size would:

- Exceed the maximum size as specified by the BLOCKS operand of DSPSERV CREATE when the data space was created.
- Extend the installation limit for the accumulated size of all data spaces. This limit is either the system default or can be set by the SYSDEF job control command.
- Be smaller than 1 or greater than 524287 blocks.

STOKEN=stoken-addr
Specifies the address of the eight-byte STOKEN for the data space (returned from DSPSERV CREATE).

BLOCKS=size | size-addr
Defines the amount of storage (in number of 4K-blocks) by which the current size of the data space is to be increased. **size-addr** specifies the address of an area where the size is specified.

Note: The amount of storage taken from VSIZE is always the number of 4K-blocks you specify for extension, rounded up to the next multiple

of 8, if necessary. For example, if you specify 10 blocks, the system rounds this number to 16 and takes 64K from VSIZE.

NUMBLKS=numblks-addr
Specifies the address of a 4-byte area where the system returns the size by which the data space was extended.

MF=S Specifies the **standard** form of the macro, which is used to place the parameters into an inline parameter list. This is also the default if the MF parameter is omitted.

MF=(L,...
Specifies the **list** form of the macro, which is used to construct a non-executable control program parameter list.

lstaddr specifies the address of the area that the system is to use for the parameter list.

attr is an optional 1- to 60-character input string which can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code **attr**, the system provides a value of 0D, which forces the parameter list to a double-word boundary.

PLISTVER=0|1 specifies which parameter list the system is to generate. 0 produces a 56-character parameter list; 1 (which is always recommended) produces a 60-character list.

No other parameters may be specified if the list form of the macro is chosen.

MF=(E,...
Specifies the **execute** form of the macro, which uses the parameter list generated by the list form of the macro.

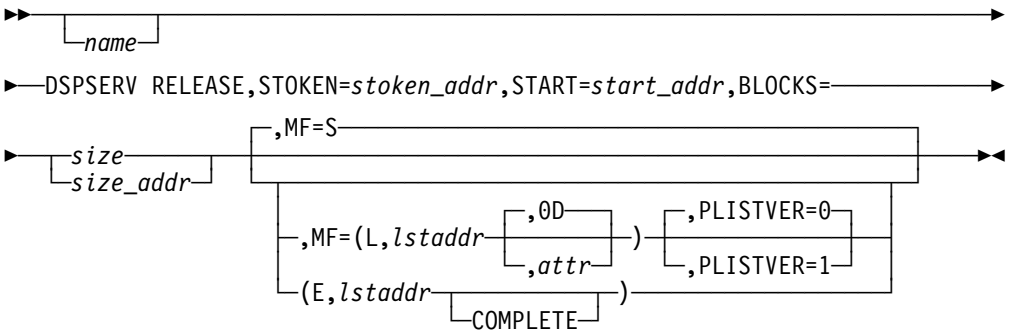
lstaddr specifies the address of the parameter list. This address must not be in a data space. If the caller of the macro is in 24-bit mode, the address of the parameter list must not be above the 16MB line.

COMPLETE specifies that the system is to check for required parameters and supply optional parameters that are not specified.

Return Codes in Register 15 (and Reason Codes in Register 0)

00 Successful completion
08 R0=X'xx050200': Extending the data space would cause the data space limits (as specified in the SYSDEF job control command) to be exceeded.
08 R0=X'xx050201': No system GETVIS storage available (page manager).
08 R0=X'xx050202': No virtual storage available (page manager).
08 R0=X'xx050203': No real storage available (page manager).

DSPSERV RELEASE (Release Data Space) Macro



Requirements for the caller:

AMODE: 31

RMODE: 24 or ANY

ASC Mode: Primary or AR (access register) if SYSSTATE ASCENV=AR

DSPSERV RELEASE requests the system to release a data space. Use this macro when you have finished using a data space or when you want to reuse it for another purpose. Releasing a data space means to initialize the virtual storage area to hexadecimal zeros and to return the resources (used to contain your data) to the system. Although the data contained in the virtual storage is discarded, the virtual storage itself remains intact and is available for further use.

A non-zero key program can release the storage of any data space it owns, provided its PSW key matches the storage key of the data space. A key-0 program can release the storage of any data space it owns and other data spaces of the caller's partition.

The system rejects the request if the released size would be outside the data space range or be zero.

STOKEN=stoken-addr
Specifies the address of the eight-byte STOKEN for the data space (returned from DSPSERV CREATE).

START=start-addr
Specifies the address of a four-byte variable that contains the start address of the block of storage to be returned to the system. The address must be on a 4KB boundary.

BLOCKS=size | size-addr
Defines either the length of the storage area (in blocks of 4K bytes) that the system is to release or the address of a field that contains this length.

MF=S Specifies the **standard** form of the macro, which is used to place the parameters into an inline parameter list. This is also the default if the MF parameter is omitted.

MF=(L,...
Specifies the **list** form of the macro, which is used to construct a non-executable control program parameter list.

lstaddr specifies the address of the area that the system is to use for the parameter list.

attr is an optional 1- to 60-character input string which can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code **attr**, the system provides a value of 0D, which forces the parameter list to a double-word boundary.

PLISTVER=0|1 specifies which parameter list the system is to generate. 0 produces a 56-character parameter list; 1 (which is always recommended) produces a 60-character list.

No other parameters may be specified if the list form of the macro is chosen.

MF=(E,...

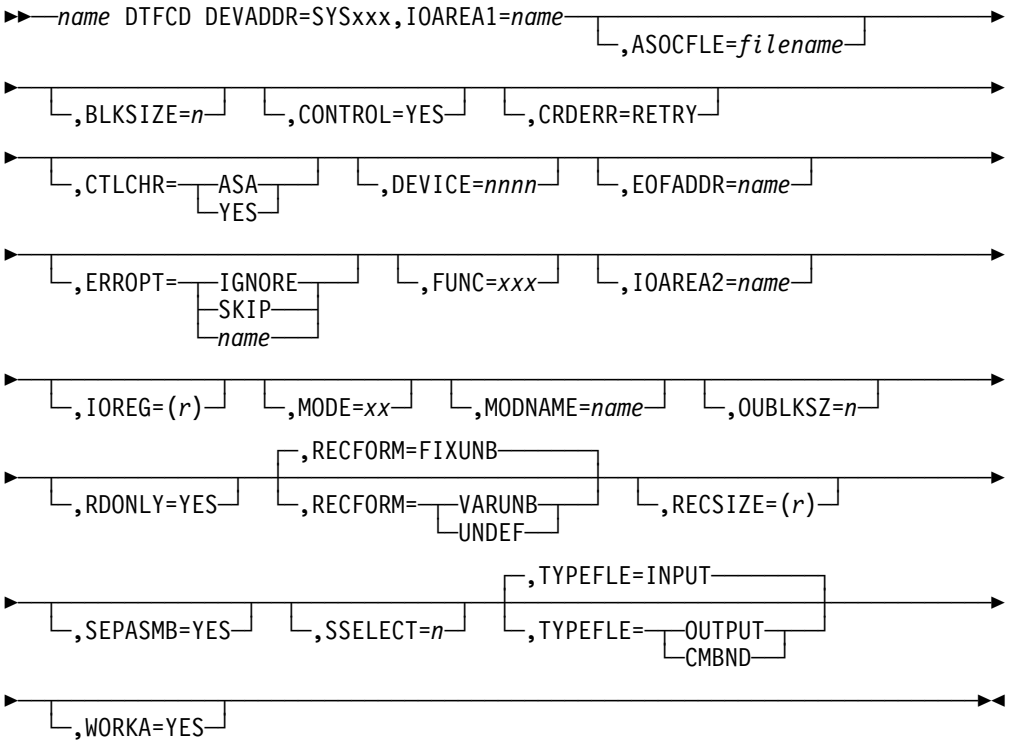
Specifies the **execute** form of the macro, which uses the parameter list generated by the list form of the macro.

lstaddr specifies the address of the parameter list. This address must not be in a data space. If the caller of the macro is in 24-bit mode, the address of the parameter list must not be above the 16MB line.

COMPLETE specifies that the system is to check for required parameters and supply optional parameters that are not specified.

The return code in register 15 is always 0.

DTFCD (Define the File for Card I/O) Macro



Required RMODE: **24**

This macro defines a file for an IBM card I/O device or an IBM 3881 Optical Mark Reader.

If not stated otherwise, the operands of the DTFCD macro can be specified for all three types of files (INPUT, OUTPUT, CMBND).

ASOCFLE=filename

This operand is used together with the FUNC operand to define associated files on the IBM 3525. For a discussion of associated files see "Programming for Associated Files" in the *VSE/ESA System Macro User's Guide*. ASOCFLE specifies the file name of associated read, punch, or print files as shown in Figure 6 on page 109. It enables macro sequence checking by the logic module of each associated file. One file name is required per DTF for an associated file.

This operand applies to input and output files.

FUNC=...	Read DTFCD	Punch DTFCD	Print DTFPR
FUNC=PW		Filename of associated print DTFPR	Filename of associated punch DTFCD
FUNC=RP	Filename of associated punch DTFCD	Filename of associated read DTFCD	
FUNC=RPW	Filename of associated punch DTFCD	Filename of associated print DTFPR	Filename of associated read DTFCD
FUNC=RW	Filename of associated print DTFPR		Filename of associated read DTFCD

Examples:

If FUNC=PW is specified, then specify:

1. The file name of the print DTFPR in the ASOCFLE=filename operand of the punch DTFCD.
2. The file name of the punch DTFCD in the ASOCFLE=filename operand of the print DTFPR.

If FUNC=RPW is specified, then specify:

1. The file name of the punch DTFCD in the ASOCFLE=filename operand of the read DTFCD.
2. The file name of the print DTFPR in the ASOCFLE=filename operand of the punch DTFCD.
3. The file name of the read DTFCD in the ASOCFLE=filename operand of the print DTFPR.

Figure 6. ASOCFLE Operand Usage with Print Associated Files

BLKSIZE=n

Enter the length of the I/O area (IOAREA1). If the record format is variable or undefined, enter the length of the largest record.

To use the IBM 3741 diskette as input device for SYSIPT data (as spooling device for VSE/POWER), you can specify a maximum value of 512 bytes for BLKSIZE.

If the operand is omitted, the defaults for the various IBM devices are as follows:

```
160  For column binary processing on the 3505 or 3525.
 96  For the 5424 and 5425.
 80  For all other devices.
```

If FUNC=I is specified for a file on the IBM 3525, the length specified for BLKSIZE must be 80 data bytes if CTLCHR=YES or if ASA is not specified; if CTLCHR=YES or if ASA is specified, the length must be 81 bytes.

CONTROL=YES

This operand is specified if a CNTRL macro is to be issued for a file. If this operand is specified, CTLCHR must be omitted.

The CNTRL macro cannot be used for an input file with two I/O areas (that is, when the IOAREA2 operand is specified), or for an input file used in association with a punch file (when the operand FUNC=RP or RPW is specified) on the 3525; in this case, however, this operand can be specified in the DTFCD for the associated punch file.

CRDERR=RETRY

This operand applies to card output on the 2520 or 2540. It specifies the operation to be performed if an error is detected. From this specification, IOCS generates a retry routine and a save area for the card punch record.

If a punching error occurs, it is usually ignored and operation continues. The error card is stacked in stacker P1 (punch), while correct cards are stacked in the stacker you select. If the CRDERR=RETRY operand is included and an error condition occurs, IOCS also notifies the operator and then enters the wait state. The operator can either cancel the job, ignore the error, or instruct IOCS to repunch the card.

CTLCHR=ASA | YES

This operand is required if first-character control is to be used on an output file. ASA denotes the American National Standards character set. YES denotes the System/370 character set. See Appendix A, “Control Character Codes” on page 429 for the complete list of codes. If this operand is specified, CONTROL must be omitted.

DEVADDR=SYSIPT | SYSPCH | SYSRDR | SYSnnn

This operand specifies the logical unit name to be associated with a file. The logical unit represents an actual I/O device address and is used in the ASSGN job control statement to assign an actual I/O device address to the file.

SYSIPT, SYSPCH, or SYSRDR must not be specified:

- For a file on an IBM 3881
- For a combined file on an IBM 2520 or 2540 (TYPEFLE=CMBND)
- For an associated file on an IBM 3525 (FUNC=RP, RW, RPW, or PW)
- If the operand FUNC=I is specified
- If the MODE operand is specified with C, O, or R.

DEVICE=nnnn

For nnnn, specify the device code of the IBM device associated with the file. The code you specify can be one of the following:

```
2520
2540
3505
3525
3881
```

If the operand is omitted, 2540 is assumed.

EOFADDR=name

This entry must be included for input and combined files; it specifies the symbolic name of your end-of-file routine. IOCS automatically branches to this routine on an end-of-file condition. In your routine, you can perform any

operations required for the end of the file (you generally issue a CLOSE instruction for the file).

IOCS detects end-of-file conditions in the card reader by recognizing the characters /* punched in card columns 1 and 2 (column 3 must be blank).

ERROPT=IGNORE | SKIP | name

This operand specifies the desired error-exit option. The operand applies to files as follows:

- Input with any of the possible specifications.
- Output with the specification IGNORE, except associated output files. For an associated output file, do not use the operand at all.

The functions of the specifications are described below:

ERROPT=IGNORE

The error is to be ignored. When control returns to your program, register 1 contains the address of the error record and, for output files, byte 3, bit 3 of the CCB is set on (see page on page 60). You can check this bit and take the appropriate action to recover from the error. Only one I/O area and no work area is permitted for output files. When IGNORE is specified for an input file associated with a punch file (FUNC=RP or RPW) and an error occurs, a PUT for the card in error must nevertheless be given for the punch file.

ERROPT=SKIP

The record in error is not to be made available for processing. The next card is read and processing continues.

ERROPT=name

IOCS branches to your routine when an error occurs. Register 1 contains the address of the record in error; register 14 contains the return address.

In your routine, you may perform whatever actions you desire. However, GET macros may not be issued for cards in the same device. If the file is an associated file, PUT macros may not be issued for cards in the same device.

If any other IOCS macros are issued in the routine, register 14 must be saved. If the operand RDONLY=YES is specified, register 13 must also be saved. At the end of your routine, return to IOCS by branching to the address in register 14. If the input file is associated with an output file (FUNC=RP, RPW, or RW), no punching or printing must be done for the card in error. IOCS continues processing by reading the next card.

Note: When ERROPT is specified for an input file and an error occurs, the /* end-of-file card may be lost. After having taken the action for the card in error as specified by the ERROPT operand, IOCS reads the next card, which is assumed to be a data card. If this card is an end-of-file card, IOCS cannot recognize the end-of-file condition.

FUNC=R | P | I | RP | RW | RPW | PW

This operand specifies the type of input or output file to be processed on the IBM 3525.

R indicates read.
P indicates punch.
W indicates write (print).

When FUNC=I is specified, the file will be both punched and interpreted; no associated file is necessary to achieve this. The information printed will be the same as the information punched, in contrast to FUNC=PW, where any relation between the information printed and the information punched is determined by your program. When FUNC=I is specified the file can have only one I/O area.

RP, RW, RPW, and PW are used, together with the ASOCFLE operand, to specify associated files. When one of these specifications is coded for a file, it must also be coded for the associated file(s). Each of the associated files can have only one I/O area.

IOAREA1=name
This operand specifies the name of the input or output area used for this file.

If the macro is issued for a combined file, this operand specifies the input area. If IOAREA2 is not specified, the area specified in this operand is used for both input and output.

IOAREA2=name
This operand specifies the name of a second I/O area. If the file is a combined file and the operand is specified, the designated area is an output area.

If this operand is specified for a file on the IBM 3881, the IOREG operand must also be specified.

This operand must not be specified if either:

- In the FUNC operand, you code one of the specifications I, RP, RPW, RW, or PW, or
- For an output file, you specify ERROPT=IGNORE.

IOREG=(r)
If two input or output areas are used instead of a work area, this operand specifies the register (any of 2 through 12) into which IOCS puts the address of the record. For output files, IOCS puts into this register the address where the user can build a record. This operand cannot be used for combined files.

This operand must be specified for a file on the IBM 3881 if the IOAREA2 operand is specified.

MODE=E | C | O | R | EO | ER | CO | CR
This operand specifies the mode used to process an input or output file on an IBM 3505 or 3525.

E = Normal EBCDIC mode, which is also the default. It is also the default if only O or R is specified.
C = Column binary mode.
O = Optical mark read (OMR) mode.
R = Read column eliminate mode.

Valid entries are:

- For a file on the IBM 3505: E, C, O, R, EO, ER, CO, and CR.
- For a file on the IBM 3525: E, C, R, ER, and CR.

- For SYSIPT, SYSPCH, or SYSRDR: E. O, and R (with or without E or C) cannot be specified for output files.

If O or R is specified (with or without E or C), a format descriptor card defining the card columns to be read, or eliminated, must be provided. See “Format Descriptor Card” in the *VSE/ESA System Macro User's Guide* for instructions on how to write this card and on how to code and process OMR data.

MODNAME=name
This operand is used to specify the name of the logic module that is used with the DTF table to process the file. If the logic module is assembled with the program, MODNAME must specify the same name as the CDMOD macro.

If this operand is omitted, standard names are generated for calling the logic module. If two DTF macros call for different functions that can be handled by a single module, only one module is called.

OUBLKSZ=n
This operand is used in conjunction with IOAREA2, but only for a combined file. Enter the maximum number of characters to be transferred at one time. If this entry is not included and IOAREA2 is specified, the same length as defined by BLKSIZE is assumed.

RDONLY=YES
This operand is specified if the DTF is used with a read-only module. Each time a read-only module is entered, register 13 must contain the address of a 72-byte doubleword-aligned save area.

Every task requires its own uniquely defined save area, and each time an imperative macro (except OPEN or OPENR) is issued, register 13 must contain the address of the save area associated with that task. Because each of the save areas is unique for a certain task, the module is reentrant. Thus, the module can be used concurrently by two or more tasks.

If an ERROPT routine issues I/O macros using the same read-only module that caused control to pass to the error routine, your program must provide another save area. One save area is used for the normal I/O operations; the second for I/O operations in the ERROPT routine. Before returning to the module that entered the ERROPT routine, register 13 must contain the save area address originally specified for the task.

If this operand is omitted, the module generated is not reenterable, and no save area is required.

RECFORM=FIXUNB | VARUNB | UNDEF
This operand specifies the record format of the file: fixed length, variable length, or undefined. If the record format is fixed unblocked (FIXUNB,) this operand may be omitted. This operand must specify FIXUNB if you also specified one of the following:

TYPEFLE=INPUT
TYPEFLE=CMBND
FUNC=I
DEVICE=3881

RECSIZE=(r)
For undefined records, this operand specifies the register (any one of 2 through 12) that contains the length of the output record. You must load the

length of each record into the specified register before you issue the PUT macro for the record.

SEPASMB=YES
Include this operand only if your DTFCD macro is to be assembled separately. This produces an object module ready to be cataloged into a suitable sublibrary by the name used as file name. The name is used as the module's transfer address. If you omit this operand, the assembler assumes that the DTFDA macro is assembled together with your program.

SSELECT=n
This operand specifies the valid stacker-select character for a file. If this operand is not specified, cards are selected into the NR (normal read) or NP (normal punch) stackers.

This operand must not be specified for:

- Combined files
- Files on the IBM 3881
- Read files associated with punch files (FUNC=RP or FUNC=RPW) on an IBM 3525.

In this case, SSELECT=n may be specified for the associated output file.

When this operand is used, the program ignores CONTROL=YES with input files.

TYPEFLE=INPUT | OUTPUT | CMBND
This operand specifies whether a file is input, output, or combined. A combined file can be specified for an IBM 2520 or for an IBM 2540 with the punch-feed-read feature. TYPEFLE=CMBND is applicable if both GETs and PUTs are issued for the same card file.

Only TYPEFLE=INPUT can be specified for the 3881. If OUTPUT or CMBND is specified, the DTF defaults to DEVICE=2540 and a non-executable CDMOD logic module is produced. The MNOTE

```
IMPROPER DEVICE. 2540 ASSUMED
```

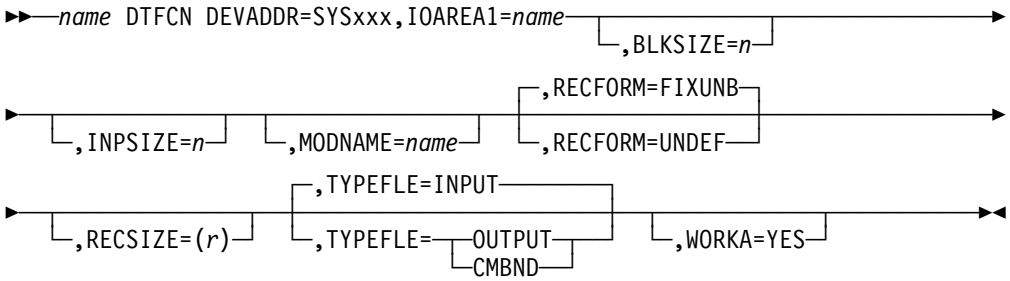
is then printed at the time of assembly.

If the operand is omitted, INPUT is assumed.

WORKA=YES
If I/O records are processed in work areas instead of in the I/O areas, specify this operand. You must set up the work area in storage. The address of the work area, or a general-purpose register which contains the address, must be specified in each GET and PUT macro.

If ERROPT=IGNORE is specified for an output file or if DEVICE=3881, WORKA=YES must not be specified.

DTFCN (Define the File for Console I/O) Macro



Required RMODE: **24**

The macro defines an input or output file that is processed on an IBM console printer-keyboard, or display operator console. DTFCN provides GET/PUT logic as well as PUTR logic for a file.

BLKSIZE=n

This operand specifies the length of the I/O area; if the PUTR macro is used (TYPEFLE=CMBND is specified), this operand specifies the length of the output part of the I/O area. For the undefined record format, BLKSIZE must be as large as the largest record to be processed. The length must not exceed 256 characters.

If the console buffering option is specified at system generation time and the device is assigned to SYSLOG, physical IOCS can increase throughput for each actual output record not exceeding 80 characters. This increase in throughput results from starting the output I/O command and returning to the program before output completion. Regardless of whether or not output records are buffered (queued on an I/O completion basis), they are always printed or displayed in first-in-first-out order.

DEVADDR=SYSLOG | SYSnnn

This operand specifies the logical unit name associated with the file. Specify DEVADDR=SYSLOG to get partition prefixes (BG, F1, F2, F3, ... Fn) for message identification.

DEVADDR=SYSLOG must be specified if your DTFCN macro includes TYPEFLE=CMBND.

Specifying DEVADDR=SYSnnn is not recommended because:

- The lines you write to the console do not have a partition identifier.
- A GET request for input from the console cannot be buffered, and the system waits for the requested input. Until the operator supplies this input, no other console communication can take place. Thus, your system's console can become a serious performance bottleneck.

INPSIZE=n

This operand specifies the length of the input part of the I/O area for PUTR macro usage.

IOAREA1=name

This operand specifies the name of the I/O area used by the file. For PUTR macro usage, the first part of the I/O area is used for output, and the second part is used for input. The lengths of these parts are specified by

the BLKSIZE and INPSIZE operands respectively. The I/O area is not cleared before or after a message is printed, or when a message is canceled and reentered on the console.

MODNAME=name
This operand specifies the name of the logic module generated by this DTFCN macro. If this entry is omitted, standard module names are generated for the logic module.

A module name must be given when two phases (each containing a DTFCN macro) are link-edited into the same program. Under such conditions, omission of this operand results in unresolved address constants.

RECFORM=FIXUNB | UNDEF
This operand specifies the record format of the file: fixed length or undefined. FIXUNB must be specified if TYPEFLE=CMBND is specified. FIXUNB is assumed if the RECFORM operand is omitted.

RECSIZE=(r)
For undefined records, this operand is required for output files and is optional for input files. It specifies a general register (2 to 12) that contains the length of the record. On output, you must load the length of each record into the specified register before you issue a PUT macro. If specified for input files, IOCS provides the length of the record transferred to storage.

TYPEFLE=INPUT | OUTPUT | CMBND
This operand specifies a file as input, output, or combined. If INPUT is specified, code is generated for both input and output files. If OUTPUT is specified, code is provided for output files only.

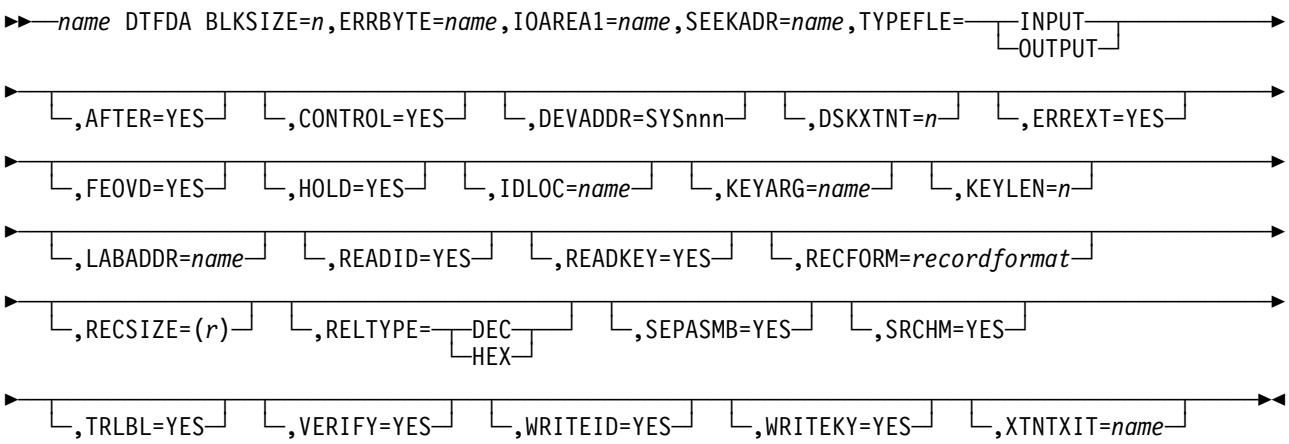
CMBND must be specified if you use the PUTR macro. This causes coding to be generated for:

- Input and output files.
- The use of PUTR macros, which ensures that a message requiring an operator response is not deleted from the console display.

When CMBND is specified, DEVADDR=SYSLOG must also be specified.

WORKA=YES
This operand indicates that a work area is used with the file. A GET or PUT macro moves the record to or from the work area. A PUTR macro moves the record from and to the work area.

DTFDA (Define the File for Direct Access) Macro



Required RMODE: **24**

The DTFDA macro defines a file for Direct Access Method (DAM) processing. If not stated otherwise, the operands of the DTFDA macro can be specified for both input and output files.

DAM does not support FBA devices.

AFTER=YES
This operand must be included for output files if any records (or an additional record) are written in a file by a formatting WRITE (count, key and data) following the last record previously written on a track. The remainder of the track is erased. That is, whenever either of the macros

```
WRITE filename,AFTER
WRITE filename,RZERO
```

is used in a program, this operand is required.

BLKSIZE=n
This operand indicates the size of the I/O area by specifying the maximum number of characters that are transferred to or from the area at any one time. When undefined, variable length or spanned records are read or written, the area must be large enough to accommodate the largest record. The chapter “I/O Area Specification” in the *VSE/ESA System Macro User's Guide* describes how to compute the size of an I/O area.

IOCS uses this specification to set up the count field of the CCW for reading or writing records.

CONTROL=YES
Include this operand if a CNTRL macro is issued for this file. The CNTRL macro for seeking on a disk allows you to specify a track address on which access movement should begin for the next READ or WRITE macro. While the arm is moving, you may process data and/or request I/O operations on other devices.

DEVADDR=SYSnnn

This operand must specify the symbolic unit (SYSnnn) associated with a file if the symbolic unit is not provided via an EXTENT job control statement. If such a unit is provided, its specification overrides the DEVADDR specification. This specification, or symbolic unit, represents an actual I/O address and is used in the ASSGN job control statement to assign the actual I/O device address to the file.

Note: EXTENT job control statements provided for DAM must be supplied in ascending order, and the symbolic units for multi-volume files must be assigned in consecutive order.

DSKXTNT=n

This operand indicates the maximum number of extents (up to 256) that are specified for a file. When this operand is used together with FIXUNB, VARUNB, or UNDEF specified in the RECFORM operand, it indicates that a relative ID is used in the SEEKADR and IDLOC locations. If DSKXTNT=n is omitted, a physical ID is assumed in the SEEKADR and IDLOC locations.

If RECFORM=SPNUNB is specified, DSKXTNT is required. If relative addressing is used, the RELTYPE operand must also be specified.

ERRBYTE=name

This operand is required. It specifies the name of a two-byte field in which IOCS stores an error-condition or status code. For description of these codes, see “Error Handling” in the *VSE/ESA System Macro User's Guide* .

ERREXT=YES

This operand enables irrecoverable I/O errors (occurring before a data transfer takes place) to be indicated to your program. This error information is indicated in the bytes named in the ERRBYTE operand and is available after the WAITF macro has been issued.

FEOVD=YES

This operand is specified if code is generated to handle end-of-volume records. It should be specified only when reading a file which was built using DTFSD and the FEOVD macro.

HOLD=YES

This operand provides for the track hold function, which is to be specified at system generation time. If the operand is omitted, the track hold function is not performed. For details, see “DASD Record Protection (Track Hold)” in the *VSE/ESA System Macro User's Guide*.

IDLOC=name

Include this operand if you want IOCS to supply the ID of a record after each READ or WRITE (ID or KEY) is completed. Specify the name of a record reference field in which IOCS is to store the ID. WAITF should be used before referencing this field. Do not specify the same field for IDLOC and SEEKADR.

Note: When the record to be read or written is the last record of the cylinder, an end-of-cylinder indication is posted in ERRBYTE1, bit 2, but the address returned is that of the **first** record of the **next** cylinder. If, in addition, the end-of-volume indication is posted, the address returned in IDLOC is all 1 bits.

IOAREA1=name
This operand must be included. It specifies the name of the input/output area used for the file. IOCS routines transfer records to or from this area. The specified name must be the same as the name used in the DS instruction that reserves this area of storage.

KEYARG=name
This operand must be included if records are identified by key; that is, if either of the macros

```
READ filename,KEY
WRITE filename,KEY
```

is used in a program, this entry and the corresponding KEYLEN operand are required. KEYARG specifies the name of the key field in which you supply the record key to IOCS.

The KEYARG operand is required for formatting WRITE (WRITE filename,AFTER) operations for files containing keys if RECFORM=VARUNB or SPNUNB. It is required also when the macro

```
READ filename,ID
```

is specified and if KEYLEN is not zero. When record reference is by key, IOCS uses this specification at assembly time to set up the data address field of the CCW for search commands.

KEYLEN=n
This operand must be included if record reference is by key or if keys are read or written. It specifies the number of bytes in each key. All keys must have the same length. If this operand is omitted, IOCS assumes a key length of zero.

If there are keys recorded on disk and this entry is absent, a WRITE ID or READ ID writes or reads the data portion of the record.

When the record reference is by key, IOCS uses this specification and your IOAREA1 specification to locate the data field in the I/O area.

LABADDR=name
You may require one or more user-standard labels in addition to the standard file label. If so, you must include your own routine to check (or write) the labels. The name of this routine is specified in this operand. IOCS branches to the routine after having processed the standard label. For more information about the handling of user-standard labels, see the section “Processing of User Labels” on page 439.

Note that the routine always gets control in 24-bit addressing mode.

READID=YES
This operand must be included for an input file if, in your program, the macro 'READ filename,ID' is used.

READKEY=YES
This operand must be included for an input file if, in your program, the macro 'READ filename,KEY' is used.

RECFORM=FIXUNB | SPNUNB | UNDEF | VARUNB
This operand specifies the type of records in the input or output file. The specifications are:

FIXUNB For fixed-length records. All records are considered unblocked. If you want blocked records, you must provide your own blocking and deblocking.

SPNUNB For spanned records. This specification is for unblocked variable-length logical records of less than 32,768 bytes per record.

UNDEF For undefined records. This specification is required only if the records are of undefined format.

VARUNB For variable-length records. This specification is for unblocked variable-length records.

For a description of record formats, see “Record Types” in the *VSE/ESA System Macro User's Guide*.

RECSIZE=(r)

This operand must be included if undefined records are specified (RECFORM=UNDEF). It specifies the number of the general-purpose register (any of 2 through 12) that contains the length of each individual input or output record.

Whenever an undefined record is read, IOCS supplies the length of the data area for that record in the specified register.

When an undefined record is written, you must load the length of the data area of the record (in bytes) into this register, before you issue the WRITE macro for the record. IOCS adds the length of the key when required.

When records are written (AFTER specified in the WRITE macro), IOCS uses the length to set up the count area written on disk. IOCS adds the length of both the count and the key when required.

RELTYPE=DEC | HEX

This operand specifies whether the zoned decimal (DEC) or hexadecimal (HEX) form of the relative ID is to be used. When FIXUNB, VARUNB, or UNDEF is specified in the RECFORM operand, RELTYPE should be supplied only if the DSKXTNT operand (relative ID) is specified.

If the operand is omitted, a hexadecimal relative ID is assumed. However, if DSKXTNT is also omitted, a physical ID is assumed in the SEEKADR and IDLOC addresses.

If RECFORM=SPNUNB is specified, the RELTYPE operand is required when relative addressing is used. If RELTYPE is omitted, a physical ID is assumed in the SEEKADR and IDLOC addresses.

SEEKADR=name

This operand must be included to specify the name of your track-reference field. In this field, you store the track location of the particular record read or written. IOCS refers to this field to determine which volume and which track contains the desired record. Whenever records are to be located by searching for a specified ID, the track-reference field must also contain the number of the record on the track.

SEPASMB=YES

Include this operand only if the DTFDA macro is to be assembled separately. This produces an object module ready to be cataloged into a suitable sublibrary by the name used as file name. The name is used as the

module's transfer address. If you omit this operand, the assembler assumes that the DTFDA macro is assembled together with your program.

SRCHM=YES
If records are identified by key, this operand may be included to cause IOCS to search multiple tracks for each specified record. The macros

```
READ filename,KEY
WRITE filename,KEY
```

cause IOCS to search the track specified in the track-reference field and all following tracks in the cylinder, until the record is found or the end of the cylinder is reached. If the file ends before the end of the cylinder and the record is not found, the search continues into the next file, if any, on the cylinder. EOC, instead of NRF, is indicated. Without SRCHM=YES, each search is confined to the specified track.

TRLBL=YES
This operand, if specified with the LABADDR operand, indicates that user standard trailer labels are to be read or written following the user standard header labels on the user label track. Both operands must be specified for trailer label processing.

TYPEFLE=INPUT | OUTPUT
This operand must be included to indicate how standard volume and file labels are to be processed. INPUT indicates that standard labels are to be read; OUTPUT indicates that standard labels are to be written.

VERIFY=YES
This operand is included if you want to check the parity of disk records after they are written. If this operand is omitted, any records written on a disk are not verified.

WRITEID=YES
This operand must be included if the disk location for writing an output record or updating an input file is specified by a record identifier; that is, whenever the macro

```
WRITE filename,ID
```

is used in the program, this operand is required.

WRITEKY=YES
This operand must be included if the disk location for writing any output record or updating an input file is specified by record key, that is, whenever

```
WRITE filename,KEY
```

is used.

XTNTXIT=name
This operand is included if you want to process label extent information. It specifies the name of your extent exit routine. Note that the routine always gets control in 24-bit addressing mode.

During an OPEN, IOCS branches to your routine after each specified extent is checked. Upon entering your routine, IOCS stores, in register 1, the address of a 14-byte field that contains the label extent information (in binary form) retrieved from the format 1 and format 3 labels. If user labels

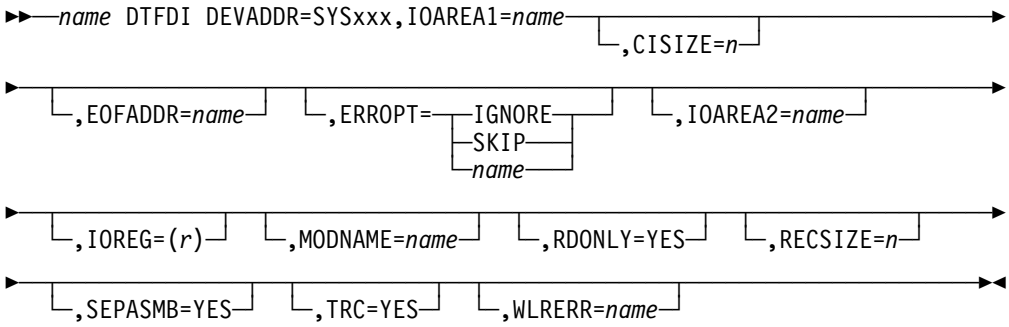
are present, the user label track is returned as a separate extent and the lower limit of the first normal extent is increased by one track. The format of this field is shown in Figure 7 on page 122.

Return to IOCS by use of the LBRET macro. Registers 2 through 13 are available in the XTNTXIT routine. Within the routine you cannot issue a macro that calls a transient routine (such as OPEN, CLOSE, DUMP, PDUMP, CANCEL, CHKPT, etc.).

Figure 7. Label Extent Information Field

Bytes	Contents
0	Extent type code (as specified in the EXTENT statement).
1	Number of extent (as determined by the EXTENT statement sequence).
2-5	Lower limit of the extent (cchh).
6-9	Upper limit of the extent (cchh).
10-11	Symbolic unit number (in hexadecimal format).
12-13	Not used.

DTFDI (Define the File for Device Independence) Macro



Required RMODE: **24**

The macro provides device independence for system logical units.

CISIZE=n This operand specifies the FBA control interval size. The value n must be an integral multiple of the FBA physical block size and, if greater than 8K, must be a multiple of 2K. The maximum value is 32768 (32K), except when assigned to SYSLST or SYSPCH, when the maximum is 30720 (30K).

If CISIZE is omitted, CISIZE=0 is assumed. For FBA devices, control interval size may be overridden for an output file at execution time by specifying the CISIZE operand of the DLBL control statement. For an input file, the CISIZE value in the format-1 label is used. If the CISIZE value is zero, then OPEN calculates a value based on the RECSIZE value specification.

DEVADDR=SYSIPT | SYSLST | SYSPCH | SYSRDR
This operand must specify the symbolic unit associated with this system file. Only the system names shown above may be specified.

EOFADDR=name
This operand specifies the name of your end-of-file routine. It is required only if SYSIPT or SYSRDR is specified. Note that the routine always gets control in 24-bit addressing mode.

IOCS branches to this routine when it detects an end-of-file condition. In this routine, you can perform any operations necessary for the end-of-file condition (you generally issue the CLOSE macro).

An end-of-file condition exists when the following occurs for SYSIPT or SYSRDR:

- For a card reader, a /* in positions 1 and 2 of the record.
- For tape, a /* in positions 1 and 2 of the record or a tapemark.
- For disk, a /* in positions 1 and 2 of the record or an end-of-file record.

IOCS detects the end-of-file condition on diskette units by recognizing that end of data has been reached on the current volume and that there are no more volumes available.

ERROPT=IGNORE | SKIP | name

This operand does not apply to output files. For output files for most devices, the job is automatically terminated after IOCS has attempted to retry writing the record.

This operand applies to wrong-length records if WLRERR is omitted. If both ERROPT and WLRERR are omitted and wrong-length records occur, IOCS ignores the error.

ERROPT specifies the function to be performed for an error block. If an error is detected when reading a magnetic tape, or a disk or a diskette volume, IOCS attempts to recover from the error. If the error is not corrected, the job is terminated unless this operand is included to specify other procedures to be taken. The three specifications are described below:

ERROPT=IGNORE

The error condition is to be ignored. The address of the error record is made available to you for processing.

ERROPT=SKIP

The error block is not to be made available for processing. The next record is read and processing continues.

ERROPT=name

IOCS is to branch to your routine when an error occurs. Register 1 contains the address of the record in error; register 14 contains the return address. Note that the routine always gets control in 24-bit addressing mode.

In your error routine, you may perform whatever functions are desired, or simply note the error condition. However, you may not issue any GET macro in the routine. If you use any other IOCS macros, you must save the contents of register 14. You must also save the contents of register 13. To access the error record, use the address in register 1; the address in the IOREG register may vary.

At the end of the error routine, return to IOCS by branching to the address in register 14. The next record is then made available for processing.

IOAREA1=name

This operand must specify the name of the input or output area used with the file. The input and/or output routines transfer records to or from this area.

If the DTFDI macro is used to define a printer file, or a card file to be processed on a printing card punch, the first byte of the output area must contain a control character.

IOAREA2=name

Two input or output areas can be allotted for a file to permit overlapped GET or PUT processing. The operand specifies the name of the second I/O area.

IOREG=(r) When two I/O areas are used, this operand specifies the general purpose register (any of 2 through 12) that points to the address of the next record. For input files, it points to the logical record available

for processing. For output files, it points to the address of the area where you can build a record.

If the operand is omitted and two I/O areas are used, register 2 is assumed.

MODNAME=name

This operand may be used to specify the name of the logic module used with the DTF table to process the file. If the logic module (DIMOD) is assembled with the program, the MODNAME operand in this DTF must specify the same name as the DIMOD macro.

If this entry is omitted, standard names are generated for calling the logic module. If two different DTF macros call for different functions that can be handled by a single module, only one standard-named module is called. The module specified by this operand is ignored if the actual device is one of the following:

A disk device
A printer of type PRT1, IBM 4248, or IBM 3800.

OPEN always provides linkage to an IBM-supplied module for these devices.

RDONLY=YES

This operand is specified if the DTF is to be used with a read-only module. Each time a read-only module is entered, register 13 must contain the address of a 72-byte doubleword-aligned save area. Each task should have its own uniquely defined save area, and each time an imperative macro (except OPEN, OPENR or LBRET) is issued, register 13 must contain the address of the save area associated with that task. The fact that the save areas are unique for each task makes the module reentrant (that is, capable of being used concurrently by several tasks).

If an ERROPT or WLRERR routine issues I/O macros using the same read-only module that caused control to pass to either error routine, the program must provide another save area. One save area is used for the initial I/O operations, and the second for I/O operations in the ERROPT or WLRERR routine. Before returning to the module that entered the error routine, register 13 must be set to the save area address originally specified for the task.

If the operand is omitted, the module generated is not reenterable and no save area need be established.

This operand is ignored for all disk devices. For these devices a read-only module is always supplied.

RECSIZE=n

This operand specifies the length of the record. For input files (SYSIPT and SYSRDR), the maximum allowable record size is 81 bytes. To ensure that control characters are handled properly during input, specify the maximum of 81 bytes (and also an I/O area of 81 bytes). In this case, the first byte of the I/O area always contains the first data byte, even if the input consists of 80 data bytes plus one control character.

For output files, RECSIZE must include one byte for control characters. The maximum length specification is 121 for SYSLST and 81 for SYSPCH.

For disk files, 121 must be specified for SYSLST, and 81 for SYSPCH. For printers and punches, DIMOD assumes a S/370-type control character if the character is not a valid ASA character. The program checks ASA control characters before S/370-type control characters. Therefore, if it is a valid ASA control character (even though it may also be a S/370-type control character), it is used as an ASA control character. Otherwise, it is used as a S/370-type control character.

Control character codes are listed in Appendix A. However:

- Stacker-selection code 3 for the IBM 2540 cannot be used if device independence is to be maintained.

If this operand is omitted, the following is assumed:

80 bytes for SYSIPT.
80 bytes for SYSRDR.
81 bytes for SYSPCH.
121 bytes for SYSLST.

SEPASMB=YES
Include this operand only if your DTFDI macro is to be assembled separately. This produces an object module ready to be cataloged into a suitable sublibrary by the name used as file name. The name is used as the module's transfer address. If you omit this operand, the assembler assumes that the DTFDI macro is assembled together with your program.

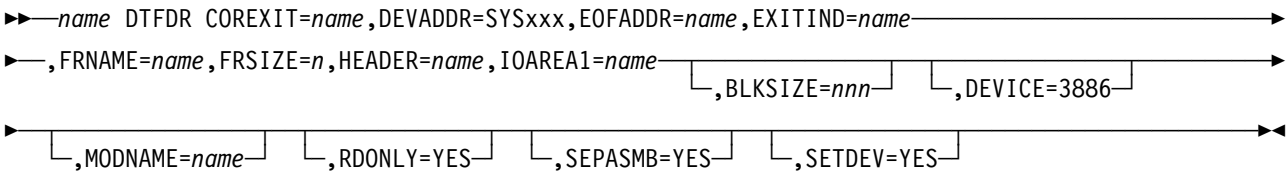
TRC=YES This operand applies to the IBM 3800 Printing Subsystem. TRC=YES specifies that a table reference character is included as the first byte of each output data line (following the optional print control character). The printer uses the table reference character (0, 1, 2, or 3) to select the character arrangement table corresponding to the order in which the table names have been specified with the CHARS operand in the SETPRT job control statement (or SETPRT macro instruction).

If the device allocated is not a printer and TRC=YES is specified, the table reference character is treated as data when a PUT is issued. If the device is a non-3800 printer, the table reference character is removed and not printed.

WLRERR=name
This operand applies only to input files on devices other than diskette units. It specifies the name of your routine to which IOCS branches if a wrong-length record is read on a tape or disk device. Note that the routine always gets control in 24-bit addressing mode.

If this operand is omitted and a wrong-length error occurs, the ERROPT routine will be invoked if it is available.

DTFDR (Define the File for Document Reader) Macro



Required RMODE: **24**

You must use the macro to define a file on an IBM 3886.

BLKSIZE=nnn
Specifies the length of the area named by the IOAREA1 keyword. The length of the area must be equal to the length of the longest record to be passed from the 3886.

If this operand is omitted, the maximum length of 130 is assumed.

Note: LIOCS does not allow you to block records read from the 3886.

COREXIT=name
Provides the symbolic name of your error correction routine. LIOCS branches to this routine whenever an error is indicated in the EXITIND byte.

You can attempt to recover from various errors that occur on the 3886 through the COREXIT routine you provide. Your COREXIT routine receives control whenever one of the following conditions occurs:

Incomplete scan
Line mark station timing mark check error
Non-recovery error
Permanent error

If any of these errors occur while the file is being opened, the COREXIT routine does not receive control and the job is canceled. Figure 8 describes normal functions for the COREXIT routine for the various error conditions; it lists the exits that must be taken from the COREXIT routine.

Error messages are provided to describe errors to the operator during program execution.

Figure 8 (Page 1 of 2). COREXIT Routine Functions

Error	Normal COREXIT Function	Exit to:
X'F1'	Do any processing that may be required. The document may have been read incorrectly; you may want to delete all data records read in (see also Note 2 on page 128 below).	Branch to the address in register 14 to return to the instruction following the macro that caused the error.
X'F2'	Eliminate the data that has been read from this document and prepare to read the next document (see also Note 1 on page 128).	Routine in your program to read the next document.

Figure 8 (Page 2 of 2). COREXIT Routine Functions

Error	Normal COREXIT Function	Exit to:
X'F3'	Rescan the line, using another format record or using image processing and editing the record in your program (see also Note 2 on page 128).	Branch to the address in reg. 14 to return to the instruction following the macro that caused the error.
X'F4' or X'F9'	Do whatever processing is necessary before the job is canceled.	Your end-of-job routine (see also Note 1 on page 128).

Notes:

1. If, in your COREXIT routine, you issue an I/O macro to the 3886 and an error occurs during that operation, control is returned to the beginning of the COREXIT routine. You must take precautions in the COREXIT routine to prevent looping in this situation. If no errors occur control returns to the instruction following the I/O macro.
2. If, in your COREXIT routine, you issue an I/O macro to the 3886, control always returns to the instruction following the macro. You should then check the completion code to determine the outcome of the operation.

DEVADDR=SYSxxx
Specifies the symbolic unit to be associated with the logical file. The symbolic unit is associated with an actual I/O device through the job control ASSGN statement.

DEVICE=3886
Indicates that an IBM 3886 is the I/O device for this file. This operand may be omitted.

EOFADDR=name
Specifies the symbolic address of your end-of-file routine. LIOCS branches to this routine whenever end of file is detected on the 3886.

EXITIND=name
Specifies the symbolic name of the 1-byte area in which the completion code is returned to the COREXIT routine for error handling from an I/O operation.

The completion codes are:

X'F0' = No errors occurred (this code should not be present when the COREXIT routine receives control).
X'F1' = Line mark station timing mark check error.
X'F2' = Non-recovery error. Do not issue the CNTRL macro to eject the document from the machine. Have the operator remove the document.
X'F3' = Incomplete scan.
X'F4' = Line mark station timing mark check and equipment check.
X'F9' = Permanent error.

Note: If any of these errors occur while the file is being opened, the COREXIT routine does not receive control and the job is canceled.

FRNAME=phasename
Specifies the name of the format record that is to be loaded when the file is opened. This name is the one you used for link-editing the desired format record. To build a format record, proceed as follows:

1. Code a DFR macro and one or more DLINT macros.
2. Assemble these macros.
3. Link-edit the assembled macros into a suitable sublibrary.

FRSIZE=n
Specifies the number of bytes to be reserved in the DTF expansion for format records. The number must equal at least the size of the largest DFR macro expansion and its associated DLINT macro expansions, plus four. This size is printed in the ninth and tenth bytes of the DFR macro expansion.

If you use the SETDEV macro in your program to change format records, you can reduce the library retrieval time by specifying a size large enough to contain all the frequently used format records. The area should then be equal to the sum of the format record sizes, plus four bytes for each format record. When the SETDEV macro is issued, the format record is loaded into this area from the related sublibrary if this record is not already in the area.

HEADER=name
Specifies the symbolic name of the 20-byte area to receive the header record from the 3886.

IOAREA1=name
Specifies the symbolic name of the input area to be used for the file. The area must be as large as the size specified in the BLKSIZE operand. If BLKSIZE is not specified, the input area must be 130 bytes.

MODNAME=name
This operand may be used to specify the name of the logic module used with the DTF table to process the file. If the logic module (DRMOD) is assembled with the program, the MODNAME operand in this DTF must specify the same name as the DRMOD macro.

If this entry is omitted, standard names are generated for calling the logic module. If two different DTF macros call for different functions that can be handled by a single module, only one standard-named module is called.

RDONLY=YES
This operand is specified if the DTF is used with a read-only module. Each time a read-only module is entered, register 13 must contain the address of a 72-byte doubleword-aligned save area. Each DTF should have its own uniquely defined save area.

Each time an imperative macro (except OPEN or OPENR) is issued for a DTF, register 13 must contain the address of the save area associated with that DTF.

If a COREXIT routine issues I/O macros using the same read-only module that caused control to pass to either error routine, your program must provide another save area. One save area is used for the normal I/O operations, and the second for I/O operations in the COREXIT routine.

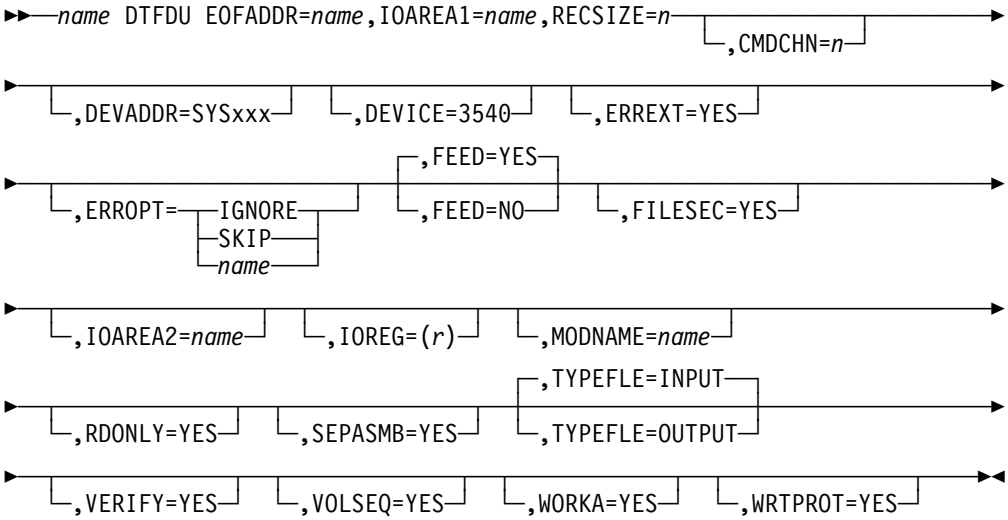
Before returning to the module that entered the COREXIT routine, register 13 must contain the save area address originally specified for that DTF.

If this operand is omitted, the module generated is not reenterable, and no save area is required.

SEPASMB=YES
Include this operand only if your DTFDR macro is to be assembled separately. This produces an object module ready to be cataloged into a suitable sublibrary by the name used as file name. The name is used as the module's transfer address. If you omit this operand, the assembler assumes that the DTFDR macro is assembled together with your program.

SETDEV=YES
Specifies that the SETDEV macro is issued in your program to load a different format record into the 3886.

DTFDU (Define the File for Diskette Unit I/O) Macro



Required RMODE: **24**

The macro defines sequential (consecutive) processing for a file contained on a diskette.

CMDCHN=n
This operand is specified to indicate the number of Read/Write CCWs to be command chained. Valid entries are 1, 2, 13, or 26; 1 is assumed if this operand is omitted. For each CCW specified by this operand, one record is processed (for example, if you code CMDCHN=13, 13 records are command chained and are processed - read or written - as a group). For entries of 2, 13, or 26, either the IOREG operand or the WORKA operand must be specified.

DEVADDR=SYSxxx
This operand specifies the symbolic unit (SYSxxx) associated with the file if an EXTENT job control statement is not provided. An EXTENT statement is not required for single-volume input files. If an EXTENT statement is provided, its specification overrides any DEVADDR specification. SYSxxx

represents an actual I/O device address, and is used in the ASSGN job control statement to assign the actual I/O device address to this file.

DEVICE=3540
This operand specifies that the file to be processed is on the IBM 3540. This operand may be omitted.

EOFADDR=name
This operand specifies the symbolic name of your end-of-file routine. IOCS automatically branches to this routine on an end-of-file condition. You can perform any operations required for the end of file in this routine (you will generally issue the CLOSE macro).

ERREXT=YES
This operand enables IOCS to indicate to your program permanent I/O errors. The operand enables your ERROPT routine to return to DUMODFx with the ERET macro. If you specify this operand, you must also specify the ERROPT operand. However, to take full advantage of this option, use the ERROPT=name operand.

ERROPT=IGNORE | SKIP | name
Specify this operand if you do not want a job to be terminated when a permanent error cannot be corrected in the diskette error routine. If attempts to reread a chain of records are unsuccessful, the job is terminated unless the ERROPT entry is included. Either IGNORE, SKIP, or the name of an error routine can be specified. The functions of these specifications are described below.

ERROPT=IGNORE
The error condition is ignored. The records are made available for processing. On output, the error condition is ignored and the records are considered written correctly.

ERROPT=SKIP
No records in the error chain are made available for processing. The next chain of records is read from the diskette, and processing continues with the first record of that chain. On output the SKIP option is the same as the IGNORE option.

ERROPT=name
IOCS branches to the routine named by this operand even if ERREXT=YES is not specified. In this routine, you can perform any function as desired or simply make note of the error condition. However, you may not issue any GET macro in the routine for records in the error chain. If you use any other IOCS macros (excluding ERET if ERREXT=YES), save the contents of register 14 and, if RDONLY=YES, also of register 13. Restore these contents to the two registers after their use.

If **ERREXT is not specified**, register 1 contains the address of the first record in the error chain. In your error routine, reference records in the error chain by referring to this address. The address in the IOREG register or the contents of the work area are variable and should not be used to process error records.

At the end of the routine, return control to IOCS by branching to the address in register 14. For a read error, IOCS skips the chain of

records in error and makes the first record of the next chain available for processing.

If **ERREXT is specified**, register 1 contains the address of a two-word parameter list:

Bytes	Contents
0-3	Address of the DTF table for the file.
4-7	The four-byte address of the first record in the error chain.

Processing is similar to that described above, except for addressing the records in error. At the end of its processing, the routine returns to LIOCS by issuing the ERET macro:

- For an input file, the program:
  - Skips the error chain and reads the next chain with an ERET SKIP.
  - Ignores the error with an ERET IGNORE.
  - Makes another attempt to read the error chain with an ERET RETRY.
- For an output file, the program:
  - Ignores the error condition with ERET IGNORE or ERET SKIP.
  - Attempts to write the error chain with an ERET RETRY. Bad spot control records (1, 2, 13, or 26 records depending on the CMDCHN specification) are written at the current diskette address, and the write chain is retried in the next 1, 2, 13, or 26 (depending on the CMDCHN specification) sectors on the disk.

Figure 9 summarizes the error options for a diskette file.

```
      Intended Processing              Your Specification

  To terminate the job ................. Nothing.

  To skip the error record ............. ERROPT=SKIP.

  To ignore the error record ........... ERROPT=IGNORE.

  To process the error record .......... ERROPT=name.

After the error record was processed:
  To leave the error-processing
  routine and

   Skip the (input) record ............. Issue ERET SKIP.

   Ignore the record ................... Issue ERET IGNORE.

   Retry reading or writing
   the record .......................... Issue ERET RETRY.
```

Figure 9. DTFDU Error Options

FEED=YES | NO

If YES is specified and IOCS detects an end-of-file condition, the diskette being processed is fed to the stacker, and a new diskette is fed to the diskette drive (provided another diskette is still in the hopper). If NO is specified, the diskette being processed is left mounted for the next job.

If the operand is omitted, YES is assumed.

FILESEC=YES

This operand applies to output only. On output it causes OPEN to set the security flag in the file label. For subsequent input, the security flag causes an operator message to be written. The operator must then reply in order to make the file available to be read.

When this operand is used with WRTPROT=YES, the reuse of the diskette is prevented.

IOAREA1=name

This operand specifies the symbolic name of the I/O area used by the file. IOCS either reads or writes records using this area. Note that you should provide an I/O area equal in size to the result obtained from multiplying the RECSIZE entry by the CMDCHN entry.

IOAREA2=name

If two I/O areas are used by GET or PUT, this operand is specified. You should provide an I/O area equal in size to the result obtained from multiplying the RECSIZE entry by the CMDCHN entry.

IOREG=(r)

This operand specifies the general purpose register (any one of 2 to 12) in which IOCS puts the address of the logical record that is available for processing. At OPEN time, for output files, IOCS puts the address of the area where the user can build a record in this register. The same register can be used for two or more files in the same program, if desired. If this is done, the problem program must store the address supplied by IOCS for each record. If this operand is specified, omit the WORKA operand.

This operand must be specified if either:

- The CMDCHN factor is 2 or higher and records are processed in one I/O area, or
- Two I/O areas are used and records are processed in both I/O areas.

MODNAME=name

This operand specifies the name of the logic module which is to process the file. If the logic module is assembled with the program, MODNAME must specify the same name as the DUMODFx macro. If this operand is omitted, standard names are generated for calling the logic module. If two DTFxx macros call for different functions that can be handled by a single module, only one module is called.

RDONLY=YES

This operand is specified if the DTF is used with a read-only module. Each time a read-only module is entered, register 13 must contain the address of a 72-byte double-word aligned save area. Each task should have its own uniquely defined save area. When an imperative macro (except OPEN, OPENR) is issued, register 13 must contain the address of the save area associated with the task. Because the save areas are unique for each task,

the module is reentrant (that is, capable of being used concurrently by several tasks).

If an ERROPT routine issues I/O macros using the same read-only module that caused control to pass to the error routine, your problem program must provide another save area. One save area is used for the normal I/O operations, and the second for input/output operations in the ERROPT routine. Before control is returned to the module that entered the ERROPT routine, register 13 must be set to the save area address originally specified for the DTF.

If this operand is omitted, the generated module is not reentrant and no save area need be established.

RECSIZE=n
This operand specifies (in bytes) the length of each record in the input/output area (1 to 128 bytes).

SEPASMB=YES
Include this operand only if your DTFDU macro is to be assembled separately. This produces an object module ready to be cataloged into a suitable sublibrary by the name used as file name. The name is used as the module's transfer address. If you omit this operand, the assembler assumes that the DTFDU macro is assembled together with your program.

TYPEFLE=INPUT | OUTPUT
This operand indicates whether the file is an input or output file.

VERIFY=YES
This operand specifies that the input on an IBM 3741/3742 must be verified before processing may continue. If VERIFY=YES is not specified, it is assumed that the input need not be verified. If VERIFY=YES is specified and the input is not verified, the job is canceled and message 4n57I is issued. If the operand is specified for an output file, it will be ignored.

VOLSEQ=YES
This operand is valid only on input. If specified, it causes OPEN to ensure that the volume sequence numbers of a multi-volume file are in ascending and sequential order. However, if the volume sequence number of the first volume processed is blank, no volume sequence checking is done.

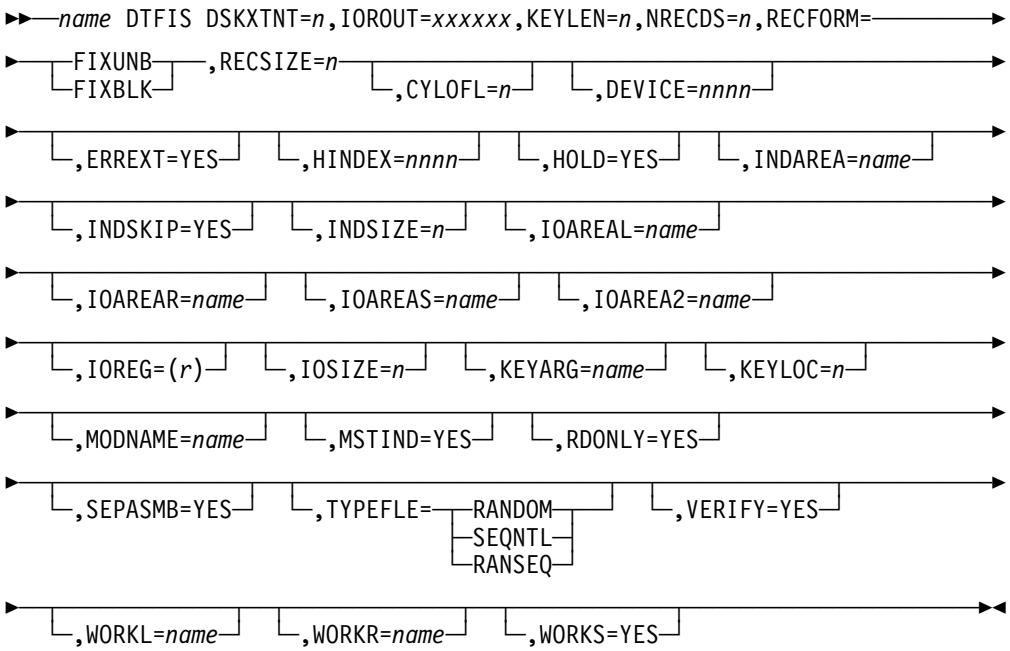
WORKA=YES
If I/O records are processed or built in work areas instead of in the I/O areas, specify this operand. You must set up the work area in storage. The address of the work area, or a general register containing the address, must be specified in each GET or PUT macro. For GET or PUT, IOCS moves the record to or from the specified work area.

When this operand is specified, the IOREG operand must be omitted.

WRTPROT=YES
This operand indicates that an output file is to be created with write-protect (which means, the file cannot be overwritten). For 3540 support, this has no effect on subsequent input processing of the file.

Note: When this operand is used with FILESEC=YES, reuse of the diskette is prevented.

DTFIS (Define the File for Indexed Sequential Access) Macro



Required RMODE: **24**

The macro defines a disk file for the Indexed Sequential Access Method.

Note: Since all the devices on which ISAM runs are no longer supported, your ISAM programs must use the *ISAM Interface Program (IIP)* to process files that have been converted from ISAM format to VSE/VSAM format. For details, see the *VSE/VSAM User's Guide and Application Programming* under “Advantages of the ISAM IIP”.

CYLOFL=n

This operand must be included if cylinder overflow areas are reserved for a file. Do not include this entry if no overflow areas are reserved.

When a file is loaded or when records are added, this operand is required to reserve the areas for cylinder overflow. It specifies the number of tracks to be reserved on each cylinder. The maximum number of tracks that can be reserved on each cylinder is:

```
For an IBM 2311              8
For an IBM 2314 or 2319     18
For an IBM 3330 or 3333     17
For an IBM 3340             10
```

DEVICE=2311 | 2314 | 3330 | 3340

This operand specifies the unit that contains the prime data area and overflow areas for the logical file. For ISAM the prime data area and the overflow areas must be on the same device type, and, for a 3340, the data modules must be of the same size (35 or 70MB).

DSKXTNT=n

This operand must be included to specify the maximum number of extents for this file. The number must include all the data area extents if more than

one disk area is used for the data records, and all the index area and independent overflow area extents that are specified by EXTENT job control statements. Thus the minimum number specified by this entry is 2: one extent for one prime data area, and one for a cylinder index. Each area assigned to an ISAM file is considered an extent.

Note: Master and cylinder indexes are treated as one area. When there is one master index extent, one cylinder index extent, and one prime data area extent, DSKXTNT=2 could be specified.

ERREXT=YES

This operand is required for IOCS to supply your program with detailed information about irrecoverable I/O errors occurring before a data transfer takes place, and for your program to be able to use the ERET imperative macro to return to IOCS specifying an action to be taken for an error condition.

Some error information is available for testing by your program after each imperative macro is executed, even if ERREXT=YES is not specified, by referencing field **filenameC**. For filename, give the name that you specified in the name field of the DTFIS macro for the file. One or more of the bits in the filenameC byte may be set to 1 by IOCS. The meaning of the bits varies depending on what was specified in the IOROUT operand; Figure 10 shows the meaning if IOROUT=ADD, RETRVE, or ADDRTR was specified; Figure 11 on page 137 shows the meaning if IOROUT=LOAD was specified.

If ERREXT=YES is not specified, IOCS returns the address of the DTF table in register 1, as well as any data-transfer error information in filenameC, after each imperative macro is executed; non-data-transfer error information is not given. After testing filenameC, return to IOCS by issuing any imperative macro except ERET; no special action is taken by IOCS to correct or check an error.

If ERREXT=YES is specified, IOCS returns the address of an ERREXT parameter list in register 1 after each imperative macro is executed, and information about both data-transfer and non-data-transfer errors in filenameC. The format of the ERREXT parameter list is shown in Figure 12 on page 138. After testing filenameC and finding an error, return to IOCS by using the ERET imperative macro; IOCS takes the action indicated by the ERET operand. If HOLD=YES (and ERREXT=YES), ERET must be used to return to IOCS to free any held track.

In your program, check bit 7 of DTF byte 16 for a block size compatibility error when adding to, or extending a file. If the block size specified in your program is not equal to the block size of the previously built file, this bit will be set to 1.

Figure 10 (Page 1 of 2). FilenameC-Status Byte if IOROUT Specifies ADD, RETRVE, or ADDRTR

Bit	Meaning if Set to 1
0	Disk error – An irrecoverable disk error has occurred (except wrong-length record.)
1	Wrong-length record – A wrong length record has been detected during an I/O operation.

Figure 10 (Page 2 of 2). FilenameC-Status Byte if IOROUT Specifies ADD, RETRVE, or ADDRTR

Bit	Meaning if Set to 1
2	End of file – End of file has been encountered during sequential retrieval.
3	No record found – The record to be retrieved has not been found in the file. This applies to RANDOM (RANSEQ) and to SETL in SEQNTL (RANSEQ) when KEY is specified, or after GKEY. This may also be a hardware error.
4	Invalid ID specified – The ID specified to the SETL in SEQNTL (RANSEQ) is outside the prime file limits.
5	Duplicate record – The record to be added to the file has a duplicate record key of another record in the file.
6	Overflow area full – An overflow area in a cylinder is full, and no independent overflow area has been specified; or an independent overflow area is full, and the addition cannot be made. You should assign an independent overflow area or extend the limit.
7	Overflow – The record being processed in one of the retrieval functions (RANDOM/SEQNTL) is an overflow record.

Figure 11. FilenameC-Status Byte if IOROUT=LOAD

Bit	Meaning if Set to 1
0	Disk error – An irrecoverable disk error has occurred (except wrong-length record.)
2	Prime area full – The next to the last track of the prime data area has been filled during the load or extension of the file. Issue the ENDFL macro, then do a load extend on the file with next extents given.
3	Cylinder-index area full – The cylinder-index area is not large enough to contain all entries needed to index each cylinder specified for the prime data area. This condition can occur during the execution of the SETFL. Extend the upper limit of the cylinder index by using a new EXTENT statement.
4	Master index full – The master index area is not large enough to contain all the entries needed to index each track of the cylinder index. This condition can occur during SETFL. Extend the upper limit, if you are creating the file, by using an EXTENT statement; or reorganize the file and assign a larger area.
5	Duplicate record – The record to be added to the file has a duplicate record key of another record in the file.
6	Sequence check – The record being loaded is not in sequential order.
7	Prime data area overflow – There is not enough space in the prime data area to write an EOF record. This condition can occur during the execution of the ENDFL macro.

Figure 12. ERREXT Parameter List

Bytes	Contents
0-3	Address of the DTF block
4-7	Virtual storage address of the record in error.
8-15	Disk address (mbbcchhr) of the error, where: **m =** Extent sequence number **r =** A record number which can be inaccurate if a read error occurred during a read of the index of the highest level.
16	Record identification: **Bit** **Meaning if 1** 1 Data record 2 Track-index record. 3 Cylinder- or master-index record. 4-5 Reserved. 6 Read operation. 7 Write operation
17	Command code of failing CCW.

HINDEX=2311 | 2314 | 3330 | 3340

This operand specifies the type of the disk unit that contains the highest index.

Placing the highest index on a separate unit is recommended only if that unit is physically separate from the unit(s) holding the track indexes and the data of the file, and if it has its own access mechanism. If this operand is omitted, 2311 is assumed.

HOLD=YES

This operand provides for the track hold option for both data and index records. If the HOLD operand is omitted, the track hold function is not performed. Because track hold cannot be performed on a LOAD file, HOLD=YES cannot be specified when IOROUT=LOAD.

If HOLD=YES and ERREXT=YES, your program must issue the ERET macro to return to the ISAM module to free any held tracks.

INDAREA=name

This operand specifies the name of the area assigned to the cylinder index. If specified, all or part of the cylinder index resides in virtual storage thereby increasing throughput. If this operand is included, INDSIZE must be included.

If the area assigned to INDAREA is large enough for all the index entries to be read into virtual storage at one time and the index skip feature (INDSKIP) is not specified, no presorting of records need be done. If the area assigned to INDAREA is not large enough, the records processed should be presorted to fully utilize the resident cylinder index.

INDSKIP=YES

When cylinder index entries reside in virtual storage, this operand specifies the index skip feature. This feature allows ISAM to skip any index entries preceding those needed to process a given key. If the index skip operand is omitted, the cylinder indexes are processed sequentially.

This operand may be specified only with the INDAREA and INDSIZE operands and increases throughput only when:

- The records are presorted.
- The allocated virtual storage is insufficient for storing all of the cylinder index.
- One or more large segments of the file are not referenced.

INDSIZE=n

This operand specifies the length (in bytes) of the index area assigned in virtual storage to the cylinder index by INDAREA. The minimum you can specify is:

```
n = (m + 3) (keylength + 6)
```

where

m =
The number of entries to be read into virtual storage at a time.

3 = The number of dummy entries.

6 = A pointer to the cylinder.

If m is set equal to the number of prime data cylinders+1, the entire cylinder index is read into virtual storage at one time. The maximum value for n = 32767.

The resident index facility is suppressed if this operand is omitted, the minimum requirement is not met at assembly time, or an irrecoverable read error is encountered while reading the index.

IOAREAL=name

This operand must be included when a file is created (loaded) or when records are added to a file. It specifies the name of the output area used for loading or adding records to the file. The specified name must be the same as the name used in the DS instruction that reserves the area of storage. The ISAM routines construct the contents of this area and transfer records to disk.

This output area must be large enough to contain the count, key, and data areas of records. Furthermore, the data-area portion must provide enough space for the sequence-link field of overflow records whenever records are added to a file (see Figure 13 on page 140).

If IOAREAL is increased to permit the reading and writing of more than one physical record on disk at a time, the IOSIZE operand must be included when records are added to the file. In this case, the IOAREAL area must be at least as large as the number of bytes specified in the IOSIZE operand.

When simultaneously building two ISAM files using two DTFs, do not use a common IOAREAL. Also, do not use a common area for IOAREAL, IOAREAR, and IOAREAS in multiple DTFs.

IOAREAR=name

This operand must be included whenever records are processed in random order. It specifies the name of the input/output area for random retrieval (and updating). The specified name must be the same as that used in the DS instruction that reserves this area of storage.

The I/O area must be large enough to contain the data area for records. Furthermore, the data-area portion must provide enough space for the sequence-link field of overflow records (see Figure 14 on page 141).

IOAREAS=name

This operand must be included whenever records are processed in sequential order by key. It specifies the name of the input/output area used for sequential retrieval (and updating). The specified name must be the same as that used in the DS instruction that reserves this area of storage.

This I/O area must be large enough to contain the key and data areas of unblocked records and the data area for blocked records. Furthermore, the data-area portion must provide enough space for the sequence-link field of overflow records (see Figure 14 on page 141).

Function	Output Area Requirements (in No. of Bytes)						
	Count		Key		Seq. Link		Data
Load unblocked records	8	+	key-length	+	0	+	R
Load blocked records	8	+	key-length	+	0	+	R x B
Add unblocked records	8	+	key-length	+	10	+	R
Add blocked records The greater of	8 8	+ +	key-length key-length	+ +	0 10	+ +	R x B R
B = Blocking factor R = Record length							

Figure 13. Output Area Requirements for Loading or Adding Records to a File by ISAM

IOAREA2=name

This operand permits overlapping of I/O with indexed sequential processing for either the load (creation) or sequential retrieval functions. Specify the name of an I/O area to be used when loading or sequentially retrieving records. The I/O area must be at least the length of the area specified by either the IOAREAL operand for the load function or the IOAREAS operand for the sequential retrieval function. If the operand is omitted, one I/O area is assumed. If TYPEFLE=RANSEQ, this operand must not be specified.

IOREG=(r)

This operand must be included whenever records are retrieved and processed directly in the I/O area. It specifies the register that ISAM uses to indicate which individual record is available for processing. ISAM puts the address of the current record in the designated register (any of 2 through 12) each time a READ, WRITE, GET, or PUT is executed.

IOROUT=LOAD | ADD | RETRVE | ADDRTR

This entry must be included to specify the type of function to be performed. The specifications have the following meanings:

IOROUT=LOAD

To build a logical file on a disk or to extend a file beyond the highest record presently in a file.

IOROUT=ADD
To insert new records into a file.

IOROUT=RETRVE
To retrieve records from a file for either random or sequential processing and/or updating.

IOROUT=ADDRTR
To both insert new records into a file (ADD) and retrieve records for processing and/or updating (RTR).

Note: The disk device must be in READ/WRITE mode for all functions.

	I/O Area Requirements (in No. of Bytes)			
	Count	Key	Seq. Link	Data
Retrieve unblocked records	0	+ *key-length	+ 10	+ R
Retrieve blocked records The greater of	0	+ 0	+ 0	+**R x B
	0	+ 0	+ 10	+ R
B = Blocking factor R = Record length	* Only for sequential retrieval ** Including keys			

Figure 14. I/O Area Requirements for Random or Sequential Retrieval by ISAM

IOSIZE=n
This operand specifies the (decimal) number of bytes in the virtual-storage area assigned for the add function using IOAREAL. The number n can be computed using the following formula:

```
n = m (keylength + blocksize + 40) + 24

Where m = The maximum number of physical records that can
          be read into virtual storage at one time.
```

The number n must be at least equal to

```
(keylength + blocksize + 74)
```

This formula accounts for a needed sequence link field for unblocked records or short blocks (see Figure 13 on page 140 and Figure 14).

If the operand is omitted, or if the minimum requirement is not met, no increase in throughput is realized.

The number n should not exceed the track capacity because throughput cannot be increased by specifying a number larger than the capacity of a track.

KEYARG=name
This operand must be included for random READ/WRITE operations and sequential retrieval initiated by key. It specifies the symbolic name of the key field in which you must supply the record key to ISAM.

KEYLEN=n
This operand must be included to specify the number of bytes in the record key.

KEYLOC=n

This operand must always be specified if RECFORM=FIXBLK. It supplies ISAM with the high-order position of the key field within the data record. That is, if the key is recorded in positions 21-25 of each record in the file, this operand should specify 21.

ISAM uses this specification to locate (by key) a specified record within a block. The key area of a block of records contains the key of the highest record in the block. To search for any other records, ISAM locates the proper block and then examines the key field within each record in the block.

MODNAME=name

This operand may be used to specify the name of the logic module used with the DTF table to process the file. If the logic module is assembled with the program, the MODNAME in the DTF must specify the same name as the ISMOD macro. If this entry is omitted, standard names are generated for calling the logic module. If two DTF macros call for different functions that can be handled by a single module, only one module is called.

MSTIND=YES

This operand is included whenever a master index is used or is to be built for a file. The location of the master index is specified by an EXTENT job control statement.

NRECDS=n

This operand specifies the number of logical records in a block (called the blocking factor). It is required only if RECFORM=FIXBLK. For FIXBLK, **n** must be greater than 1; for FIXUNB, **n** must be =1.

RDONLY=YES

This operand is specified if the DTF is used with a read-only module. Each time a read-only module is entered, register 13 must contain the address of a 72-byte doubleword-aligned save area. Each task should have its own uniquely defined save area. Register 13 must contain the address of the save area associated with the task each time an imperative macro (except OPEN, OPENR, LBRET, SETL, or SETFL) is issued. The fact that the save areas are unique for each task makes the module reentrant (that is, capable of being used concurrently by several tasks).

RECFORM=FIXUNB | FIXBLK

This operand specifies whether records are blocked or unblocked. FIXUNB is used for unblocked records, and FIXBLK for blocked records. If FIXBLK is specified, the key of the highest record in the block becomes the key for the block and must be recorded in the key area.

The specification that is included when the logical file is loaded onto a disk must also be included whenever the file is processed.

Records in the overflow area(s) are always unblocked, but this has no effect on this operand. RECFORM refers to records in the prime data area only.

RECSIZE=n

This operand must be included to specify the number of characters in the data area of each individual record. This operand should specify the same number for additions and retrieval as indicated when the file was created.

SEPASMB=YES

Include this operand only if your DTFIS macro is to be assembled separately. This produces an object module ready to be cataloged into a suitable sublibrary by the name used as file name. The name is used as the module's transfer address. If you omit this operand, the assembler assumes that the DTFIS macro is assembled together with your program.

TYPEFLE=RANDOM | SEQNTL | RANSEQ

This operand must be included when IOROUT=RETRVE or IOROUT=ADDRTR. The operand specifies the type(s) of processing performed by your program for the file.

RANDOM is used for random processing. Records are retrieved in random order specified by key.

SEQNTL is used for sequential processing. Your program specifies the first record retrieved, and thereafter ISAM retrieves records in sequential order by key. The first record is specified by key, ID, or the beginning of the logical file (see “SETL (Set Limits) Macro” on page 367).

RANSEQ is used if both random and sequential processing are to be performed for the same file. If RANSEQ is specified, the IOAREA2 operand must not be specified.

TYPEFLE is not required for loading or adding functions.

VERIFY=YES

Use this operand if you want to check the parity of disk records after they are written. If this operand is omitted, any records written on a disk are not verified.

WORKL=name

This operand must be included whenever a file is created (loaded) or records are added to a file. It specifies the name of the work area in which you must supply the data records to ISAM for loading or adding to the file. The specified name must be the same as the name used in the DS instruction that reserves this area of storage.

This work area must provide space for one logical record when a file is created (for blocked records: data; for unblocked records: key and data).

The original contents of WORKL are changed due to record shifting in the ADD function.

WORKR=name

When records are processed in random order, this operand must be included if the individual records are to be processed in a work area rather than in the I/O area. It specifies the name of the work area. This name must be the same as the name used in the DS instruction that reserves this area of storage. This area must provide space for one logical record (data area). When this entry is included and a READ (or WRITE) macro is executed, ISAM moves the individual record to (or from) this area.

WORKS=YES

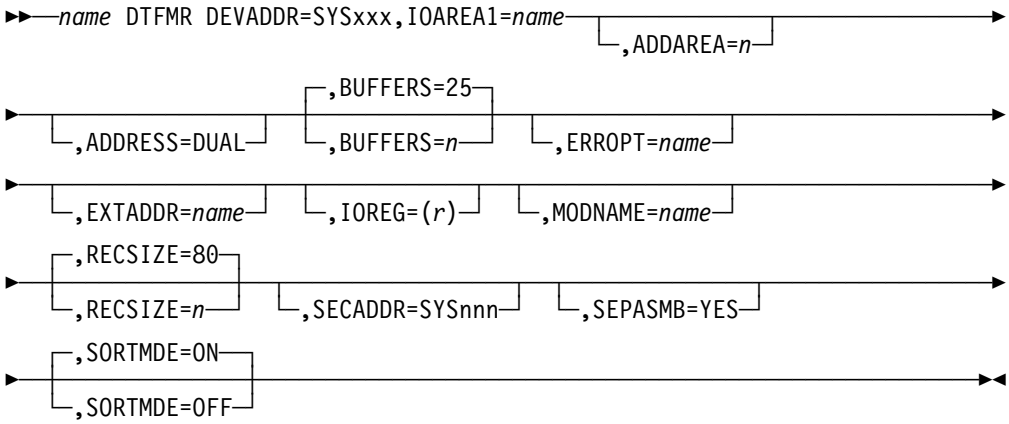
When records are processed in sequential order, this operand must be included if the individual records are processed in work areas rather than in the I/O area. Each GET and PUT macro must specify the name of the work area to or from which ISAM is to move the record. When processing unblocked records, the area must be large enough for one record (data

area) and the record key (key area). For blocked records, the area must be large enough for one logical record (data area) only. The work area requirements are as shown in Figure 15.

Function	Unblocked Records	Blocked Records
Load The greater of	10 K + D	10 D
Add The greater of	10 K + D	10 K + D
Random Retrieve	D	D
Sequential Retrieve	K + D	K + D

Figure 15. Work Area Requirements

DTFMR (Define the File for Magnetic Reader Input) Macro



Required RMODE: **24**

DTFMR defines an input file processed on an IBM 1255 or 1419 magnetic character reader, or an IBM 1270 or 1275 optical character reader/sorter.

ADDAREA=n
This operand must be included only if an additional buffer work area is needed. For n, specify the number of additional bytes you desire in each buffer. The sum of the ADDAREA and RECSIZE specifications must not exceed 250. This area can be used as a work area and/or output area and is reset to binary zeros when the next GET or READ for the file is executed.

ADDRESS=DUAL
This operand must be included only if the 1419 or 1275 contains the dual address adapter. If the single address adapter is used, this operand must be omitted.

BUFFERS=25 | n
This operand is included to specify the number of buffers in the document buffer area. The limits for n are 12 and 254. 25 is assumed if this operand is omitted.

DEVADDR=SYSxxx
This operand is required and specifies the symbolic unit to be associated with the file. The symbolic unit represents an actual I/O device address used in the ASSGN job control statement to assign the actual I/O device address to the file.

ERROPT=name
This operand may be included only if the CHECK macro is used. For name, give the name of the routine that the CHECK macro branches to if any error condition is posted in byte 0, bits 2 to 4 (and bit 5, if no control address is specified in the CHECK macro) of the buffer status indicators. It is your responsibility to exit from this routine (see the “CHECK (Check I/O Completion) Macro” on page 69).

EXTADDR=name
This operand specifies the name of your stacker selection routine. The routine receives control if an external interrupt occurs while documents are

being read or being sorted internally. You may omit this if you specify SORTMDE=OFF.

IOAREA1=name
This operand is required and specifies the name of the document buffer area that will be used by the file. Figure 3 on page 70 shows the format of the document buffer area.

IOREG=(r)
This operand specifies the general-purpose register (one of 2 to 12) that the IOCS routines and your routines use to indicate which individual document buffer is available for processing. IOCS puts the address of the current document buffer in the specified register each time a GET or READ is issued.

The same register may be specified in the IOREG entry for two or more files in the same program, if desired. In this case, your program may need to store the address supplied by IOCS for each record.

Register 2 is assumed if this operand is omitted.

MODNAME=name
This operand specifies the name of the logic module generated by MRMOD. If the operand is omitted, IOCS generates the standard system module name.

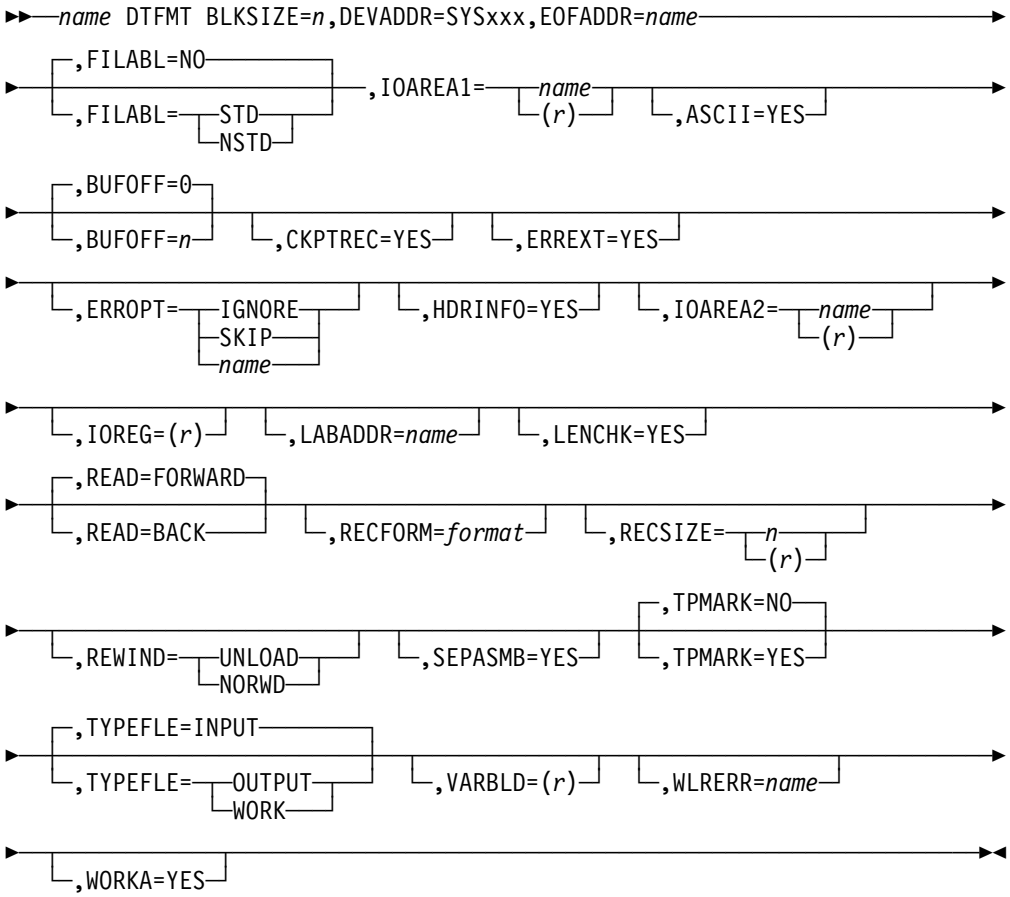
RECSIZE=80 | n
This operand specifies the actual length of the data portion of the buffer. The record size specified must be the size of the largest record processed. If this operand is omitted, a record size of 80 is assumed. The sum of the ADDAREA and RECSIZE specifications must not exceed 250.

SECADDR=SYSnnn
This operand specifies the symbolic unit to be associated with the secondary control unit address if the IBM 1275 or 1419 with the dual address adapter and LITE macro are used. Omit the operand if the pocket LITE macro is not being used.

SEPASMB=YES
Include this operand only if your DTFMR macro is to be assembled separately. This produces an object module ready to be cataloged into a suitable sublibrary by the name used as file name. The name is used as the module's transfer address. If you omit this operand, the assembler assumes that the DTFMR macro is assembled together with your program.

SORTMDE=ON | OFF
This operand specifies the method of sorting done on the 1419. SORTMDE=ON indicates that the program sort mode is being used. SORTMDE=OFF indicates that sorting is under control of the magnetic character reader. If the operand is omitted, the program sort mode is assumed.

DTFMT (Define the File for Magnetic Tape I/O) Macro



Required RMODE: **24**

The macro defines a magnetic tape file.

If not otherwise stated, the operands of the DTFMT macro can be specified for all three types of files (input, output, or work).

You need not code an MTMOD logic module. It is automatically loaded into the SVA at IPL time and linked to the problem program during OPEN processing for the DTFMT.

ASCII=YES
This operand specifies that processing of ASCII tapes is required (see Appendix B, “American National Standard Code for Information Interchange” on page 433). If this operand is omitted, EBCDIC processing is assumed. ASCII=YES is not permitted for work files.

BLKSIZE=n
For n, specify the length of the I/O area in number of bytes. If the record format is variable or undefined, give the length of the largest block of records. For a file of variable-length records, the value must include the block and record descriptor bytes.

If a READ or WRITE macro specifies a length greater than n for work files, the record to be read or written is truncated to fit into the I/O area.

The maximum block size is 32,767 bytes. The minimum size of a physical tape record (gap to gap) is 12 bytes.

For output processing, the minimum physical record length is 18 bytes. If the specified value is less than 18 (but not less than 12), IOCS does one of the following, depending on the format of the records of the file:

- **Fixed and variable length** – IOCS writes the records padded up to a length of 18 bytes as follows:
  - For EBCDIC tapes (ASCII=YES is not specified) – X'800000...'.
  - For ASCII tapes (ASCII=YES is specified) – X'5F5F5F...'.

  Note, however, that when your data records are exactly 18 bytes long and end with the padding character, they will be truncated during input processing. You can avoid this by increasing your records by one byte.
- **Spanned** – IOCS ignores your specification and assumes a specification of BLKSIZE=18.

For ASCII tapes, the BLKSIZE includes the length of any block prefix or padding characters present. If ASCII=YES and BLKSIZE is less than 18 bytes (for fixed-length records only) or greater than 2048 bytes, an MNOTE is generated because this length violates the limits specified by American National Standards Institute, Inc.

BUFOFF=0 | n

This operand can be included only when ASCII=YES is specified; it is not allowed for work files. The operand indicates the length of the block prefix. Supply this length if processing of the block prefix is required. The contents of this field are not passed on to you. For n, you can specify a value as follows:

Value	Condition
0 to 99	If TYPEFLE=INPUT
0	IF TYPEFLE=OUTPUT and the file contains records of fixed length.
4	If TYPEFLE=OUTPUT and the file contains records of variable length. In this case, the program automatically inserts the physical record length in the block prefix.

CKPTREC=YES

This operand is necessary if an input tape has checkpoint records interspersed among the data records. IOCS bypasses any checkpoint records encountered. This operand must not be included when ASCII=YES.

DEVADDR=SYSRDR | SYSIPT | SYSLST | SYSPCH | SYSnnn

This operand specifies the symbolic unit to be associated with the file. An ASSGN job control statement assigns an actual channel and unit number to the unit. The ASSGN job control statement contains the same symbolic name as DEVADDR. When processing ASCII tapes, you must specify a programmer logical unit (SYSnnn).

EOFADDR=name

This operand specifies the name of your end-of-file routine. IOCS automatically branches to this routine on an end-of-file condition. This entry

must be specified for input and work files. Note that the routine always gets control in 24-bit addressing mode.

In your routine, you can perform any operations required for the end of file (generally you issue the CLOSE macro for the file). IOCS detects end-of-file conditions in magnetic tape input by reading a tapemark and EOF when standard labels are specified. If standard labels are not specified, IOCS assumes an end-of-file condition when the tapemark is read, or, if the unit is assigned to SYSRDR or SYSIPT, when a /* is read. You must determine, in your routine, that this actually is the end of the file.

ERREXT=YES

This operand enables IOCS to indicate to your program any irrecoverable I/O errors other than tape read data checks. The operand enables your ERROPT routine to return to IOCS by means of the ERET (error return) macro.

Specifying this operand is meaningful only if you supply an error routine (by ERROPT=name). ERREXT=YES and ERROPT=name are both required for an output file (TYPEFLE=OUTPUT).

ERROPT=IGNORE | SKIP | name

This operand specifies functions to be performed when a tape read data check or (when ERREXT=YES is specified) a tape write check (irrecoverable I/O error) occurs.

The functions of these specifications are:

ERROPT=IGNORE

The error condition is completely ignored, and the records are made available for processing. When spanned records are processed, IOCS returns to your program the entire spanned record or a block of spanned records rather than just the one physical record in which the error occurred.

On output, the error is ignored and the physical record containing the error is treated as a valid record. The remainder, if any, of spanned record segments are written, if possible.

ERROPT=SKIP

On input, no records in the error block are made available for processing. The next block is read from tape, and processing continues with the first record of that block. The error block is included in the block count. When reading spanned records, the entire spanned record or a block of spanned records is skipped rather than just one physical record.

On output, the error is ignored and the physical record containing the error is treated as a valid record. The remainder, if any, of the spanned record segments are written.

ERROPT=name

This operand and ERREXT=YES are both required for an output file (TYPEFLE=OUTPUT). IOCS branches to the routine named by this operand even if ERREXT=YES is not specified. Note that the routine always gets control in 24-bit addressing mode. In your routine, you can perform any function as desired or simply make note of the error condition. However, you may not issue any GET macro in the routine

for the tape file. If you use any other IOCS macros (excluding ERET if ERREXT=YES), save the contents of register 14 and, if RDONLY=YES, also of register 13. Restore these contents to the two registers after their use.

If **ERREXT is not specified**, register 1 contains the address of the block in error. In your error routine, reference the error block by referring to this address. The address in the IOREG register or the contents of the work area are variable and should not be used to process error records.

At the end of the routine, return control to IOCS by branching to the address in register 14. For a read error, IOCS skips the error block and makes the next block of records available for processing.

If **ERREXT is specified**, register 1 contains the address of a two-word parameter list:

Bytes	Contents
0-3	Address of the DTF table for the file. Test the data transfer bit (bit 2 of byte 2). If the bit is 1, the block in error has not been read or written. If the bit is 0, data was transferred.
4-7	The 4-byte address of the first record in the error chain. For an ASCII tape, this is the address of the first logical record following the block prefix.

Processing is similar to that described above, except for addressing the error block.

At the end of its processing, the routine returns to LIOCS either by branching to the address in register 14 or, for an input file, by issuing the ERET macro with SKIP or with IGNORE. Do not use the ERET macro to return to IOCS from your routine for a tape output file.

FILABL=NO | STD | NSTD

This operand specifies what type of labels are to be processed. Specify:

NO To indicate no labels.
STD To indicate IBM- or user-standard labels.
NSTD To indicate non-standard labels.

You must furnish a routine to check or build user- or non-standard labels. Define the entry point of this routine in the LABADDR operand of the DTFMT macro for your file.

FILABL=NSTD is not permitted for ASCII files (that is, when ASCII=YES). Labels and tape data are assumed to be in the same mode.

HDRINFO=YES

This operand, if specified with FILABL=STD, causes IOCS to print standard header label information (fields 3-10) on SYSLOG each time a file with standard labels is opened. It also prints the file name, logical unit, and device address each time an end-of-volume condition is detected. Both FILABL=STD and HDRINFO=YES must be specified for header label information to be printed.

IOAREA1=name | (r)
This operand specifies the name of the I/O area. When variable-length records are processed, the size of the I/O area must include four bytes for the block size. If you use register notation, you can get the required storage from the partition GETVIS area via the GETVIS macro. This operand does not apply to work files.

IOAREA2=name | (r)
This operand specifies the name of a second I/O area. When variable-length records are processed, the size of the I/O area must include four bytes for the block size. If you use register notation, you can get the required storage from the partition GETVIS area via the GETVIS macro. This operand does not apply to work files.

IOREG=(r)
This operand specifies the register in which IOCS places the address of the logical record that is available for processing if:

- Two input or output areas are used.
- Blocked input or output records are processed in the I/O area.
- Variable unblocked records are read.
- Undefined records are read backwards.
- Neither BUFOFF=0 nor WORKA=YES is specified for ASCII files.

For output files, IOCS places, in the specified register, the address of the area where you can build a record. Any of registers 2 to 12 may be specified.

This operand cannot be used if WORKA=YES.

LABADDR=name
Enter the symbolic name of your routine to process user-standard or non-standard labels. Note that the routine always gets control in 24-bit addressing mode.

For ASCII tapes, this operand may be used only for writing and checking user standard labels that conform to American National Standards Institute, Inc. standards. Non-standard labels are not permitted.

This operand does not apply to work files.

For more information about the handling of user labels, see the section “Processing of User Labels” on page 439.

LENCHK=YES
This operand applies only to ASCII tape input if BUFOFF=4 and RECFORM=VARUNB or VARBLK. It must be included if the block length (specified in the block prefix) is to be checked against the physical record length. If the two lengths do not match, the action taken is the same as described under the WLRERR operand, but the WLR bit (byte 5, bit 1) in the DTF is not set.

READ=FORWARD | BACK
This operand specifies, for input and work files, the direction in which the tape is read. If READ=BACK is specified and a wrong-length record smaller than the I/O area is encountered, the record is read into the I/O area right-justified.

If READ=BACK is specified, REWIND=NORWD must also be specified; otherwise the tape will be repositioned after CLOSE.

RECFORM=format
This operand specifies the type of EBCDIC or ASCII records in the input or output file. For format, specify one of the following:

FIXUNB	For fixed-length unblocked records (the default)
FIXBLK	For fixed-length blocked records
VARUNB	For variable-length unblocked records
VARBLK	For variable-length blocked records
SPNBLK	For spanned variable-length blocked records (EBCDIC only)
SPNUNB	For spanned variable-length unblocked records (EBCDIC only)
UNDEF	For undefined records

Work files may use only FIXUNB or UNDEF.

On an IBM 9346 tape device, you cannot process a multi-volume file with spanned records. Your program will be canceled (unit check with command reject).

RECSIZE=n | (r)
For fixed-length blocked records, RECSIZE is required. It specifies the number of characters in each record.

When processing spanned records, you must specify RECSIZE=(r) where r is a register that contains the length of each record.

For undefined records, this entry is required for output files but is optional for input files. It specifies a general register (any of 2 to 12) that contains the length of the record. On output, you must load the length of each record into the register before you issue a PUT macro.

Spanned-record output requires a minimum record length of 18 bytes. A physical record less than 18 bytes is padded with binary zeros to complete the 18-byte requirement. This applies to both blocked and unblocked records. If specified for input, IOCS provides the length of the record transferred to virtual storage. This operand does not apply to work files.

REWIND=UNLOAD | NORWD
If this specification is not included, tapes are automatically rewound to load point, but not unloaded, on an OPEN or OPENR or a CLOSE or CLOSER macro or on an end-of-volume condition. If other operations are desired for a tape input or output file, specify:

REWIND=UNLOAD
To rewind the tape on an OPEN and to rewind and unload on a CLOSE or on an end-of-volume condition.

REWIND=NORWD
To prevent rewinding the tape at any time. This option positions the read/write head between the two tapemarks that indicate the end-of-file condition.

SEPASMB=YES
Include this operand only if your DTFMT macro is to be assembled separately. This produces an object module ready to be cataloged into a suitable sublibrary by the name used as file name. The name is used as the

module's transfer address. If you omit this operand, the assembler assumes that the DTFMT macro is assembled together with your program.

TPMARK=YES | NO
If a tapemark is desired for an output file and nonstandard labels are indicated (FILABL=NSTD), specify TPMARK=YES. If TPMARK=NO is specified together with FILABL=STD, the former specification is ignored. If FILABL=NO is specified or the FILABL operand is omitted, TPMARK=YES must be specified for IOCS to write a tapemark ahead of the first data record. The default is NO.

TYPEFLE=INPUT | OUTPUT | WORK
Use this operand to indicate whether the file is used for input or output. If INPUT is specified, the GET macro is used. If OUTPUT is specified, the PUT macro is used. If WORK is specified, the READ/WRITE, NOTE/POINTx, and CHECK macros are used.

The specification of WORK in this operand is not permitted for ASCII files.

On an IBM 9346 tape you cannot process a work file that requires previously stored data to be overwritten. Your program will be canceled (unit check with command reject).

VARBLD=(r)
This entry is required whenever variable-length blocked records are built directly in the output area (no work area is specified). It specifies the number (r) of a general-purpose register (any of 2 to 12) that always contains the length of the available space remaining in the output area.

IOCS calculates the space still available in the output area, and supplies it to you in the VARBLD register after the PUT macro is issued for a variable-length record. You can then compare the length of the next variable-length record with the available space to determine whether the record will fit in the remaining area. This check must be made before the record is built. If the record does not fit, issue a TRUNC macro to transfer the completed block of records to the tape. The current record is then built as the first record of the next block.

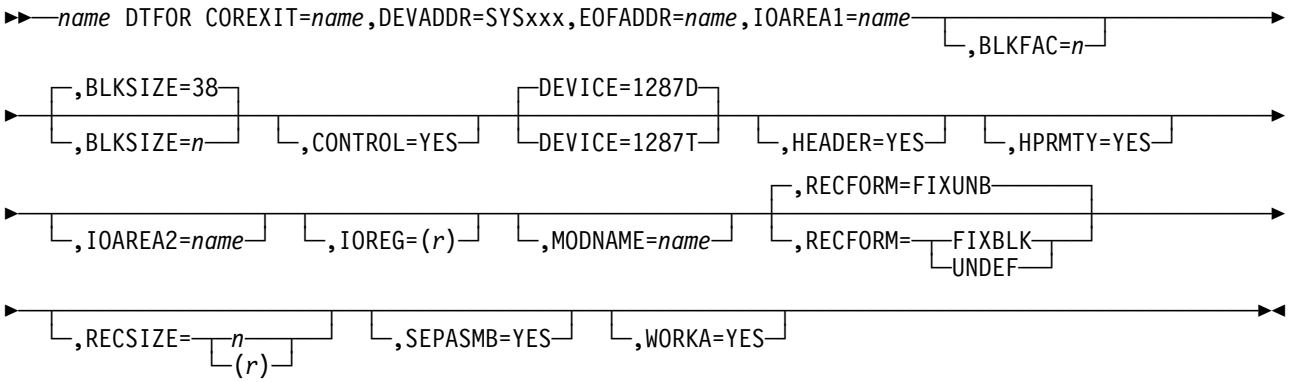
WLRERR=name
This operand applies only to tape input files. It specifies the name of your routine to receive control if a wrong-length record is read. Note that the routine always gets control in 24-bit addressing mode. If the WLRERR entry is omitted but a wrong-length record is detected by IOCS, one of the following conditions results:

- If the ERROPT operand is included for this file, the wrong-length record is treated as an error block, and handled according to your specifications for an error (IGNORE, SKIP, or name of error routine).
- If the ERROPT entry is not included, IOCS assumes the IGNORE option of ERROPT.

WORKA=YES
If I/O records are processed in work areas instead of in the I/O areas, specify this operand. You must set up the work areas in virtual storage. The symbolic address of the work area, or a general-purpose register containing the address, must be specified in each GET or PUT. Omit IOREG if this

operand is included. WORKA=YES is required for spanned record processing. It does not apply to work files.

DTFOR (Define the File for Optical Reader Input) Macro



Required RMODE: **24**

This macro is used to define an input file to be processed on an IBM 1287 Optical Reader or 1288 Optical Page Reader. If not stated otherwise, the operands of the DTFOR macro can be specified for any file on these devices.

The macro cannot be used for a file on the IBM 3881 Optical Mark Reader; use the DTFCD macro instead.

BLKFAC=n

On an IBM 1287, undefined journal tape records are processed with greater throughput when this operand is included. BLKFAC specifies the blocking factor (n) that determines the number of lines read as a block of data by one physical read. Deblocking is accomplished automatically by IOCS when the GET macro is used. The BLKFAC operand is not used with RECFORM=FIXBLK, because the blocking factor is determined from the BLKSIZE and RECSIZE operands. If the operand is included for FIXBLK, FIXUNB, or document processing, the operand is noted (in an MNOTE) and ignored.

BLKSIZE=38 | n

This operand indicates the size of the input area specified by IOAREA1. 38 is the default. For journal tape processing, BLKSIZE specifies the maximum number of characters that can be transferred to the area at any one time.

When undefined journal tape records are read, the area must be large enough to accommodate the longest record to be read if the BLKFAC operand is not specified. If the BLKFAC operand is specified, the BLKSIZE value must be determined by multiplying the maximum length that must be accommodated for an undefined record by the blocking factor desired. A BLKSIZE value smaller than this results in truncated data.

If two input areas are used for journal tape processing (IOAREA1 and IOAREA2), the size specified in this entry is the size of each I/O area.

CONTROL=YES

This entry must be included if a CNTRL macro is issued for a file. A CNTRL macro issues orders to the optical reader to perform non-data operations such as line marking, stacker selecting, and document incrementing.

COREXIT=name

COREXIT provides an exit to your error correction routine for the 1287 or 1288. Note that the routine always gets control in 24-bit addressing mode. After a GET, WAITF, or CNTRL macro is executed (to increment or eject and/or stacker select a document), an error condition causes an error correction routine to be entered with an error indication provided in filename+80.

The byte at filename+80 indicates the condition that occurred while the last line or field was read. Therefore, have your program test the byte also after any of the following macros: DSPLY, RESCN, RDLNE, CNTRL READKB, and CNTRL MARK. More than one error condition may be present. The conditions are indicated by the setting of bits as follows:

X'01' = A data check has occurred. Five read attempts for journal tape processing or three read attempts for document processing were made.

X'02' = The operator corrected one or more characters from the keyboard (DEVICE=1287T) or a hopper empty condition (see HPRMTY=YES operand) has occurred (DEVICE=1287D).

X'04' = A wrong-length record condition has occurred (for journal tapes, five read attempts were made; for documents, three read attempts were made). Not applicable for undefined records.

X'08' = An equipment check resulted in an incomplete read (ten read attempts were made for journal tapes or three for documents). If an equipment check occurs on the first character in the record, when processing undefined journal tape records, the RECSIZE register contains zero, and the IOREG (if used) points to the rightmost position of the record in the I/O area. You should test the RECSIZE register before moving records from the work area or the I/O area.

X'10' = An irrecoverable error occurred.

X'20' = End of page (EOP) occurred while records are read (in unformatted mode) from a file on the IBM 1288. Normally, on an EOP indication, the problem program ejects and stacker selects the document. After one of the macros CNTRL ESD, CNTRL SSD, CNTRL EJD in your COREXIT routine, a late stacker selection condition occurred. For the 1287, a stacker select was given after the allotted elapsed time and the document was put in the reject pocket.

X'40' = The scanner of your IBM 1287 was unable to locate the reference mark. For journal tapes, ten read attempts were made; for documents, three read attempts were made.

The action in your error correction routine depends on the requirements of your program:

- If you issue I/O macros to any device other than IBM 1287 or 1288, you must save registers 0. 1, 14, and 15 when your routine receives control. Your program must restore these registers before exiting.
- If I/O macros (other than the GET, WAITF, and READ, which cannot be used in COREXIT) are issued to your IBM 1287 and/or 1288, you must save registers 14 and 15 and later restore these registers before exiting.

All exits from the routine should be to the address specified in register 14. This provides a return to the point from which the branch to COREXIT occurred.

If the command chain bit is on in the READ CCW for which the error occurred, IOCS completes the chain upon return from the COREXIT routine.

Note: Do not issue a GET, READ, OPEN, or WAITF macro to your IBM 1287 or 1288 in the error-correction routine. Do not process records in that routine. The record which caused the exit to the error routine is available for processing on return of control to the mainline program. Any processing included in the error routine would be duplicated after return to the mainline program.

When processing journal tapes, an irrecoverable error (torn tape, tape jam, and so on) normally requires that the tape be completely reprocessed. In this case, your routine must not branch to the address in register 14 from the COREXIT routine or a program loop will occur. Following an irrecoverable error:

- The optical reader file must be closed.
- The condition causing the non-recovery must be cleared.
- The file must be reopened before processing can continue.

If an irrecoverable error occurs while processing documents (indicating, for example, a jam during an increment for a document, a scanner control failure, or an end-of-page condition), the document should be removed either manually or by nonprocess runout. In such cases, your program should branch to read the next document.

If the scanner of the device is unable to locate the document reference mark, the document cannot be processed. In this case, the document must be ejected and stacker selected before attempting to read the following document or a program loop will result.

Whenever an irrecoverable error occurs, your COREXIT routine must not branch to the address in register 14 to return to IOCS. Instead, the routine should ignore any output resulting from the document.

Eight binary error counters are used to accumulate totals of certain device-error conditions. Each of these counters occupies four bytes, starting at filename+48 (where filename is the name you specified in the name field). The error counters are listed in the table below.

Counter	Address	Description of Count
1	filename+48	Equipment check (see Note, below).
2	filename+52	Equipment check after ten read attempts for journal tapes or three read attempts for documents (see Note, below).

Counter	Address	Description of Count
3	filename+56	Wrong-length records (not applicable for undefined records).
4	filename+60	Wrong-length record error after five read attempts for journal tapes or three read attempts for documents (not applicable for undefined records).
5	filename+64	Keyboard corrections (journal tape only).
6	filename+68	Journal tape lines (including retried lines) or document fields (including retried fields) in which data checks are present.
7	filename+72	Lines marked (journal tape only).
8	filename+76	Count of total lines read from journal tape or the number of CCW chains executed during document processing.

Note: Counters 1 and 2 apply to equipment checks that result from incomplete reads or from the inability of the scanner to locate a reference mark (when processing documents only).

The counters contain binary zeros at the start of each job step. You may list the contents of these counters for analysis at end of file, or at end of job, or you may ignore the counters. To list these contents convert them from binary to a printable format.

DEVADDR=SYSnnn

This operand specifies the logical unit (SYSnnn) to be associated with the file. The logical unit represents an actual I/O device address used in the ASSGN job control statement to assign the actual I/O device address to this file.

DEVICE=1287D | 1287T

This operand specifies the I/O device associated with this file. 1287D specifies a document file. 1287T specifies a journal tape file on the IBM 1287.

From this specification, IOCS sets up the device-dependent routines for this file. For document processing you must code the CCWs.

If this operand is omitted, 1287D is assumed.

EOFADDR=name

This operand specifies the name of your end-of-file routine. IOCS automatically branches to this routine on an end-of-file condition. Note that the routine always gets control in 24-bit addressing mode.

When reading data from documents, you can recognize an end-of-file condition by pressing the end-of-file key on the console when the hopper is empty. When processing journal tapes on a 1287, you can detect an end of file by pressing the end-of-file key after the end of the tape is sensed.

When IOCS detects an end-of-file condition, it branches to your routine specified by EOFADDR. You must determine whether the current roll is the last roll to be processed when handling journal tapes. Regardless of the situation, the tape file must be closed for each roll within your EOF routine. If the current roll is not the last, OPEN must be issued. The OPEN macro allows header (identifying) information to be entered at the reader keyboard

and read by the processor when using logical IOCS. The same procedure can be used for 1287 processing of multiple journal tape rolls, as well as the method described under 'OPEN Macro' in the section 'Imperative Macros'.

HEADER=YES

This operand cannot be used for 1288 files. This operand is required if the operator is to key in header (identifying) information from the 1287 keyboard. The OPEN routine reads the header information only when this entry is present. If the entry is not included, OPEN assumes no header information is to be read. The header record size can be as large as the BLKSIZE entry and is read into the high-order positions of IOAREA1.

HPRMTY=YES

This operand is included (for the 1287D or 1288) if you want to be informed of the hopper empty condition. This condition occurs when a READ is issued and no document is present, and is recognized at WAITF time. When a hopper empty condition is detected, your COREXIT routine is entered with X'02' stored in filename+80.

This operand should be used when processing documents in the time-dependent mode of operation, which allows complete overlapping of processing with reading. See the appropriate IBM 1287 device manuals for processing details. With this method of processing, specifying HPRMTY=YES allows you to check for a hopper empty condition in your COREXIT routine. You can then select into the proper hopper the previously ejected document before return from COREXIT (via register 14).

IOAREA1=name

This operand is included to specify the name of the input area used by the file. When opening a file and before each journal tape input operation to this area, the designated area is set to binary zeros and the input routines then transfer records to this area. For document processing, the area is cleared only when the file is opened.

IOAREA2=name

A second input area can be allotted only for a journal tape file (on a 1287T). This permits an overlap of data transfer and processing operations. The specified second I/O area is set to binary zeros before each input operation to this area occurs.

IOREG=(r)

This operand specifies a general-purpose register (any one of 2 to 12) that IOCS uses to indicate the beginning of records for a journal tape file. The same register may be specified in the IOREG operand for two or more files in the same program, if desired. In this case, your program may need to store the address supplied by IOCS for each record. Whenever this operand is included for a file, the WORKA operand must be omitted, and a GET macro for the file may not specify a work area.

A read by an optical reader is accomplished by a backward scan. This places the rightmost character in the record into the rightmost position of the I/O area and subsequent characters in sequence from right to left. The register defined by IOREG points to the leftmost position of the record.

MODNAME=name

This operand may be used to specify the name of the logic module used with the DTF table to process the file. If the logic module (ORMOD) is assembled with the program, the MODNAME operand in this DTF must specify the same name as the ORMOD macro.

If this entry is omitted, standard names are generated for calling the logic module. If two different DTF macros call for different functions that can be handled by a single module, only one standard-named module is called.

RECFORM=FIXUNB | FIXBLK | UNDEF

This operand specifies the type of records in an optical reader file. One of the following may be specified:

FIXUNB For fixed-length unblocked records (default).
FIXBLK For fixed-blocked records in journal tape mode.
UNDEF For undefined records.

RECSIZE=n | (r)

For **fixed-length unblocked** records, do not specify this operand.

For **fixed-length blocked** records (journal tape mode), include this operand to specify the number, n, of characters in an individual record. The input routines use this number to deblock records, and to check the length of input records. If this operand is omitted, IOCS assumes unblocked records of fixed length.

For **undefined** journal tape records, this operand specifies the number (r) of the general-purpose register in which IOCS provides the length of each input record. For undefined document records, RECSIZE contains only the length of the last field of a document read by the CCW chain that you supply. Any one of registers 2 through 12 may be specified.

If the operand is omitted, IOCS uses register 13.

Note: When processing undefined records in document mode, you gain complete usage of the register normally used in the RECSIZE operand. You can do this by ensuring that the suppress-length-indication (SLI) flag is always on when processing undefined records.

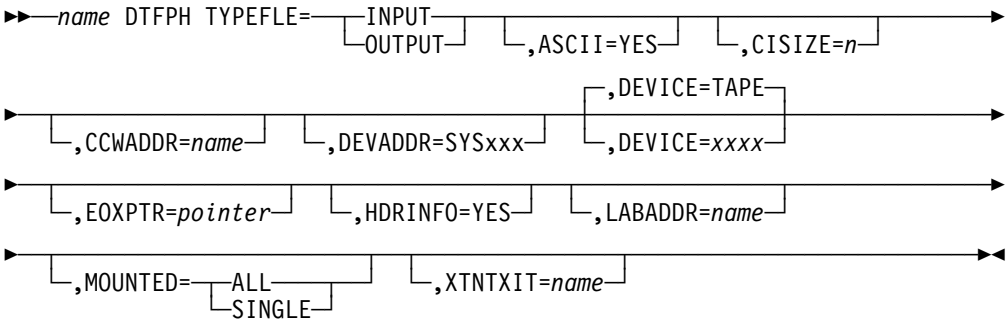
SEPASMB=YES

Include this operand only if your DTFOR macro is to be assembled separately. This produces an object module ready to be cataloged into a suitable sublibrary by the name used as file name. The name is used as the module's transfer address. If you omit this operand, the assembler assumes that the DTFOR macro is assembled together with your program.

WORKA=YES

Input records from a journal tape can be processed in work areas instead of in the input areas. If this is planned, the operand WORKA=YES must be specified, and you must set up the work area in storage. The symbolic name of the work area, or a general-purpose register containing the address of the work area, must be specified in each GET macro. When GET is issued, IOCS left-justifies the record in the specified work area. Whenever this operand is included for a file, the DTFOR IOREG operand must be omitted.

DTFPH (Define the File for Physical I/O) Macro



Required RMODE: **24**

When physical IOCS macros (EXCP, WAIT, etc.) are used in a program, disk, diskette, or tape files with standard labels need to be defined by the DTFPH macro (DTFxx macro for a file handled by physical IOCS). DTFPH must also be used for a checkpoint file on a disk.

Figure 16 shows which of the DTFPH entries can or must be coded to define a checkpoint file on disk.

Operand	Optional	Required
CCWADDR=name	X	
CISIZE=n	X	
DEVADDR=SYSnnn	X	
DEVICE=DISK		X
LABADDR=name	X	
MOUNTED=SINGLE		X
TYPEFLE=OUTPUT		X

Figure 16. Operands to Define a Checkpoint File on Disk

ASCII=YES
This operand is required to process ASCII tape files (see Appendix B, “American National Standard Code for Information Interchange” on page 433). If this operand is omitted, EBCDIC processing is assumed.

CCWADDR=name
This operand allows you to use the CCB generated within the first 16 bytes of the DTFPH table. CCWADDR specifies the symbolic name of the first CCW used with the CCB generated within the DTFPH macro. This name must be the same as the name specified in the assembler CCW statement that defines the CCW.

If you omit the operand, the location counter value of the CCB-CCW table address constant is substituted for the CCW address.

CISIZE=n
This operand specifies the FBA control-interval size. The value n must be an integral multiple of the FBA physical block size and, if greater than 8K,

must be a multiple of 2K. The maximum value is 32 768 (32K) except when assigned to SYSLST or SYSPCH, when the maximum is 30 720 (30K).

If CISIZE is omitted, CISIZE=0 is assumed. For an output file on an FBA devices, the control-interval size may be overridden at the time of program execution. You do this by specifying the CISIZE operand of the DLBL job control statement. For an input file, the CISIZE value in the format-1 label is used.

DEVADDR=SYSxxx

This operand must specify the logical unit (SYSxxx) associated with the file if a logical unit is not provided via an EXTENT job control statement. If a logical unit is provided, its specification overrides a DEVADDR specification. This specification, or logical unit, represents an actual I/O address, and is used in the ASSGN job control statement to assign the actual I/O device address to this file.

If SYSLST or SYSPCH are used as output tape units and alternate tape switching is desired upon detecting a reflective spot, the SEOV macro must be used (see “SEOV (System End-of-Volume) Macro” on page 364). When processing ASCII tape files, the only valid specification is a programmer logical unit (that is, SYSnnn).

DEVICE=TAPE | xxxx

Code the proper device identification, which may be one of the following:

TAPE If the file resides on a tape mounted onan IBM tape drive supported by VSE. For an ASCII file, TAPE is the only valid specification in this operand. TAPE is the default if you omit the operand.

DISK If the file may reside on a disk of any type, CKD or FBA. If you specify DISK, IOCS determines the disk device type when the file is opened.

3540 If the file resides on a diskette.

nnnn Which is a disk device-type code.

There is no need for you to specify a disk device type code; specify DEVICE=DISK instead. The assembler accepts the following type specification:

```
3380
```

EOXPTR=pointer

This operand is valid only if TYPEFLE=OUTPUT and MOUNTED=SINGLE is specified.

The operand points to a 4-byte field that contains the address of your end-of-extent exit routine. The routine receives control if, during OPEN processing for an output file, IOCS cannot find an additional extent. Note that the routine always gets control in 24-bit addressing mode.

On entry to the exit routine, register 15 is set to zero.

HDRINFO=YES

This operand causes IOCS to print standard header label information (fields 3-10) on SYSLOG each time a file with standard labels is opened. Likewise, the file name, symbolic unit, and device address are printed each

time an end-of-volume condition is detected. If HDRINFO=YES is omitted, no header or end-of-volume information is printed.

LABADDR=name

This operand does not apply to diskette input/output units.

You may require one or more disk or tape labels in addition to the standard file labels. If so, you must include your own routine to check (on input) or build (on output) your label(s). Specify the symbolic name of your routine in this operand. IOCS branches to this routine after the standard label is processed. Note that the routine always gets control in 24-bit addressing mode.

LABADDR may be included to specify a routine for your header or trailer labels as follows:

- Disk input or output: header labels only.
- Tape input or output: header and trailer labels.

Thus, if LABADDR is specified, your header labels can be processed for an input/output disk or tape file, and your trailer labels can be built for a tape output file. Physical IOCS reads input labels and makes them available to you for checking; it writes output labels after they are built. This is similar to the functions performed by logical IOCS.

If physical IOCS macros are used for a tape file, an OPEN must be issued for the new volume. This causes IOCS to check the HDR1 label and provides for your checking of user standard labels, if any.

When physical IOCS macros are used and DTFPH is specified for standard tape label processing, FEOV must not be issued for an input file.

For more information about the handling of user labels, see the section “Processing of User Labels” on page 439.

MOUNTED=ALL | SINGLE

This operand does not apply to diskette input/output units.

This operand must be included to specify how many extents (areas) of the file are available for processing when the file is initially opened. This operand must not be specified for tape.

Specify ALL if all extents are available for processing. When a file is opened, IOCS checks all labels on each disk pack and makes available all extents specified by your control statements. Only one OPEN is required for the file. ALL should be specified whenever you plan to process records in a manner similar to the direct access method.

After an OPEN is performed, you must be aware that the symbolic unit address of the first volume containing the file is in bytes 30 and 31 of the DTFPH table rather than in the CCB. Therefore, place this symbolic address into bytes 6 and 7 of the associated CCB before you issue an EXCP against this CCB in your program.

Specify SINGLE if only the first extent on the first volume is available for processing. SINGLE should be specified when you plan to process records in sequential order. IOCS checks the labels on the first pack and makes the first extent specified by your control statements available for processing. You must keep track of the extents and issue a subsequent OPEN

whenever another extent is required for processing. You will find the information in the DTFPH table helpful in keeping track of the extents.

The contents of the table are:

Bytes	Contents
0-15	CCB (symbolic unit has been initialized in the CCB).
54-57	Upper extent limits (cchh). For an FBA disk, the extent upper limit is the number of the first block of the last CI. If the number of blocks per CI is greater than 1, the upper extent limit can differ from the format-1 label and your specification in the DTFPH macro.
58-59	Seek address. For a disk it must be zero.
60-63	Lower extent limit (cchh for CKD).

On each OPEN after the first, IOCS makes available the next extent specified by the control cards. When you issue a CLOSE for an output file, the volume on which you are currently writing records is indicated, in the file label, as the last volume for the file.

TYPEFLE=INPUT | OUTPUT

This operand must be included to specify the type of file: input or output.

XTNTXIT=name

This operand does not apply to diskette input/output units.

Include this operand if you want to process label extent information. It specifies the symbolic name of your extent routine. The DTFPH operand MOUNTED=ALL must also be specified for the file. Note that the routine always gets control in 24-bit addressing mode.

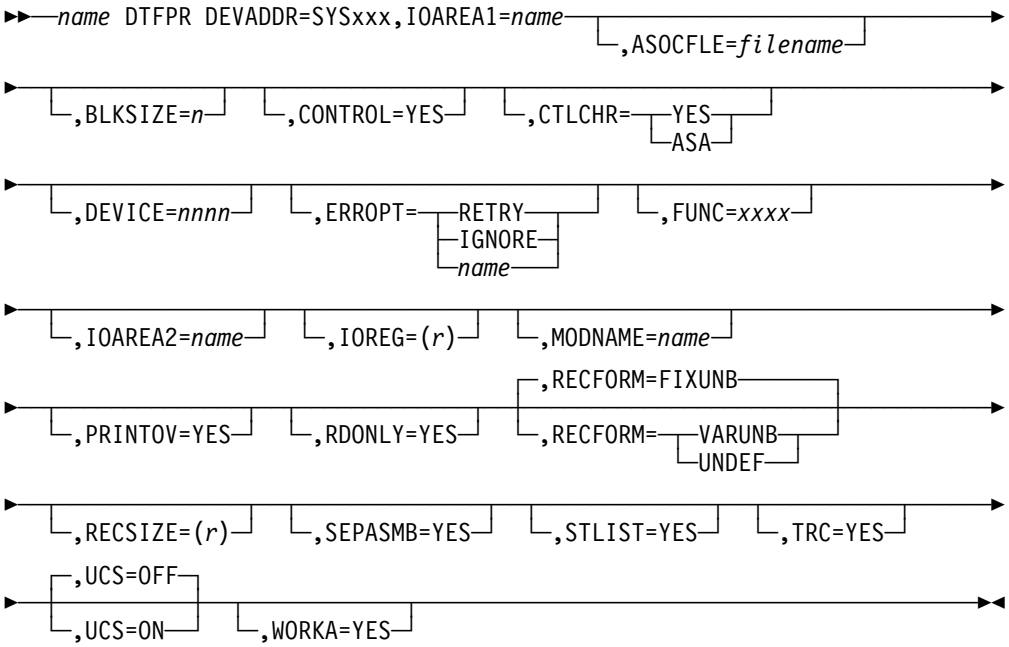
Whenever XTNTXIT is included, IOCS branches to your routine during the initial OPEN for the file. It branches after each specified extent is completely checked and after conflicts, if any, have been resolved.

When your routine receives control, register 1 contains the address of a 14-byte area from which you can retrieve label extent information (in binary form). The layout of this area is:

Bytes	Contents
0	Extent type code.
1	Extent sequence number.
2-5	Lower limit of the extent.
6-9	Upper limit of the extent.
10-11	Symbolic unit.
12	Set to zero.
13	Reserved.

Return to IOCS by using the LBRET macro.

DTFPR (Define the File for Printer) Macro



Required RMODE: **24**

The macro is used to define an output file for a printer.

ASOCFLE=filename

This operand is used together with the FUNC operand to define associated files for the IBM 3525. For a discussion of associated files see "Programming for Associated Files" in the *VSE/ESA System Macro User's Guide*.

The operand specifies the file name of an associated read and/or punch file. The specification enables macro sequence checking by the logic module of each associated file. One file name is required per DTF for associated files.

Figure 6 on page 109 shows which file name is to be specified by the ASOCFLE operand for each of the associated DTFs.

BLKSIZE=n

This operand specifies the length of IOAREA1. The maximum values which may be specified in this operand and the lengths assumed when it is omitted are given for the different devices in Figure 17 on page 165.

The actual size of the block may exceed the maximum length given in Figure 17 if the following is true:

1. Your DTFPR includes CTLCHR=YES or ASA.
2. The control character of your record is X'5A' to indicate that this is a composed page data stream (CPDS) record.
3. The specified record format is VARUNB (for variable unblocked) or UNDEF (for undefined).

Data with a control character of X'5A' can have a length of up to 32 767 bytes.

IBM Device	Maximum Length (See Note 1)	Assumed Length (See Note 2)
PRT1	512	121
1403-2	132	121
1403-3	132	121
1403-6	120	121
1403-7	120	121
1403-8	132	121
1403-9	132	121
3203	132	121
3525	64	64
3800/3200 without TRC (See Note 3)	384	136

Notes:

1. RECFORM is FIXUNB or UNDEF and operand CTLCHR is not specified. If the CTLCHR operand is specified, add one byte to the maximum value which can be specified. Add four bytes to the maximum value if RECFORM=VARUNB is specified.
2. The operand BLKSIZE=n is omitted.
3. For a 3800, the maximum length is 385 if TRC=YES is used; the assumed length is 137.

Figure 17. Maximum and Assumed Lengths for the IOAREA1 in Number of Bytes

CONTROL=YES

This operand should be specified if the CNTRL macro is issued for the file. You may omit this operand if:

1. Your CNTRL macros request immediate printer operations only, and
2. The device being used is a PRT1 printer or an IBM 4248.

Examples of immediate printer operations are: immediate space or skip; enable or disable horizontal copying.

CTLCHR=YES | ASA

Specify this operand if first-character control is used. CTLCHR=YES specifies the S/370 character set (see Appendix A, “Control Character Codes” on page 429 for a list of codes).

CTLCHR=ASA specifies the American National Standards Institute, Inc. character set. As an addition to the ASA character set, X'5A' indicating a 'composed page data stream' (CPDS) record is accepted as a valid ASA character.

If this operand is specified, omit CONTROL.

If CTLCHR=ASA is specified for a file on the IBM 3525, the + character is not allowed. To print on the first line of a card, you must issue either a

space 1 command or a skip to channel 1 command. For a print associated file on the IBM 3525, you must issue a space 1 command to print on the first line of a card.

DEVADDR=SYSLOG | SYSLST | SYSnnn
This operand specifies the symbolic unit to be associated with the printer. SYSLOG and SYSLST must not be specified for the IBM 3525.

DEVICE=nnnn
This operand specifies the type of IBM device used for the file. Specify one of the following type codes:

```
PRT1     3211
1403     3525
3203     3800
```

PRT1 refers to a 3211 or 3211-compatible IBM printer as listed under "Device Type Codes" in the *VSE/ESA System Control Statements* manual; it refers also to an IBM 4248 printer operating in native mode. Change your specification (in an existing program) to PRT1 if you want your program to:

1. Direct its print output to an IBM 4248.
2. Make use of certain IBM 4248 specific functions.

Reassemble and relink the program after this change.

ERROPT=RETRY | IGNORE | name
This operand specifies the action to be taken in the case of an equipment error. The actions you can specify are described below:

ERROPT=RETRY
Applies only if you specify also DEVICE=PRT1.

RETRY indicates that, if an equipment check with command retry is encountered, the command is retried once. If the retry is unsuccessful, a message is issued and the job is canceled.

ERROPT=IGNORE
Can be specified only for the 3525. IGNORE indicates that the error is to be ignored. The address of the record in error is put in register 1 and made available for processing. Byte 3, bit 3 of the CCB is also set on (see page on page 60); you can check this bit and take the appropriate action to recover from the error. IGNORE must not be specified for files with two I/O areas or a work area.

ERROPT=name
Applies only if you specify also DEVICE=PRT1.

If an equipment check with command retry is encountered, the command is retried once. If the retry is unsuccessful a message is issued and the job canceled.

For other types of errors (for these, see "CCB Communication Bytes" on page 58), IOCS:

1. Issues an error message.
2. Places error information into the CCB.
3. Returns control to your error routine.

In your routine, you may perform whatever actions are desired, but you should not issue any imperative macro instruction for the file invoking

the error exit. Note that the routine always gets control in 24-bit addressing mode.

To continue processing at the end of the routine, return to IOCS by branching to the address in register 14.

FUNC=W | WT | RW | RWT | RPW | RPWT | PW | PWT

This operand specifies the type of file to be processed by the IBM 3525.

W indicates print, R indicates read, P indicates punch, and T (for the 3525 only) indicates an optional 2-line printer.

RW|RWT|RPW|RPWT, and PW|PWT are used, together with the ASOCFLE operand, to specify associated files; when one of these specifications (without T) is used for a printer file, it must be specified also for the associated file(s).

Note: Do not use T for associated files, it is valid only for printer files.

If a 2-line printer is not specified for the 3525, multi-line print is assumed. T is ignored if CONTROL or CTLCHR is specified.

IOAREA1=name

This operand specifies the name of the output area.

IOAREA2=name

This operand specifies the name of a second output area.

IOREG=(r)

If two output areas and no work areas are used, this operand specifies the register into which IOCS will place the address of the area where you can build a record. For (r) specify one of the registers 2 to 12.

MODNAME=name

This operand may be used to specify the name of the logic module that is used with the DTF table to process the file. If the logic module is assembled with the program, MODNAME must specify the same name as the PRMOD macro. If this operand is omitted, standard names are generated for calling the logic module. If two DTF macros call for different functions that can be handled by a single module, only one module is called.

The module specified by this operand is ignored if the actual IBM device is one of the following:

PRT1 printer
4248 printer operating in native mode
3800 printer

OPEN always provides an IBM-supplied logic module for these devices.

PRINTOV=YES

This operand is specified if the PRTOV macro is included in your program.

RDONLY=YES

This operand is specified if the DTF is used with a read-only module. Each time a read-only module is entered, register 13 must contain the address of a 72-byte doubleword-aligned save area. Every task requires its own uniquely defined save area. Each time an imperative macro (except OPEN or OPENR) is issued, register 13 must contain the address of the save area associated with the task. Because the the save area is unique for each

task, the module is reentrant; that is, capable of being used concurrently by several tasks.

If an ERROPT routine issues I/O macros which use the same read-only module that caused control to pass to either error routine, your program must provide another save area. One save area is used for the normal I/O, and the second for I/O operations in the ERROPT routine. Before returning to the module that entered the ERROPT routine, register 13 must be set to the save area address originally specified for the task.

If this operand is omitted, the module generated is not reenterable and no save area need be established.

RECFORM=FIXUNB | UNDEF | VARUNB

The operand RECFORM=FIXUNB is specified whenever the record format is fixed. When the record format is FIXUNB, this entry may be omitted.

The entry RECFORM=UNDEF is specified whenever the record format is undefined. If the output is variable and unblocked, enter VARUNB.

RECSIZE=(r)

This operand specifies the general register (any one of 2 to 12) that will contain the length of an output record of undefined format. The length of every record must be loaded into the register before issuing the PUT macro.

SEPASMB=YES

Include this operand only if the DTFPR macro is to be assembled separately. This produces an object module ready to be cataloged into a suitable sublibrary by the name you used as file name. The name is used as the module's transfer address. If you omit the operand, the assembler assumes that the DTFPR macro is assembled together with your program.

STLIST=YES

Include this operand if the selective tape listing feature (IBM 1403 only) is used. If this entry is specified, the CONTROL, CTLCHR, and PRINTOV entries are not valid and are ignored if specified. If you specify this operand, you must specify also RECFORM=FIXUNB.

TRC=YES

This operand applies if DEVICE=3800 is specified. Specify TRC=YES if each output data line includes a table reference character following the optional print control character. The printer uses the table reference character to select the character arrangement table corresponding to the order in which the table names were specified (in the CHAR operand of the SETPRT job control statement or a SETPRT macro).

If a device code other than a 3800 is specified in the DEVICE operand, any table reference character sent to that printer is treated as data.

UCS=OFF | ON

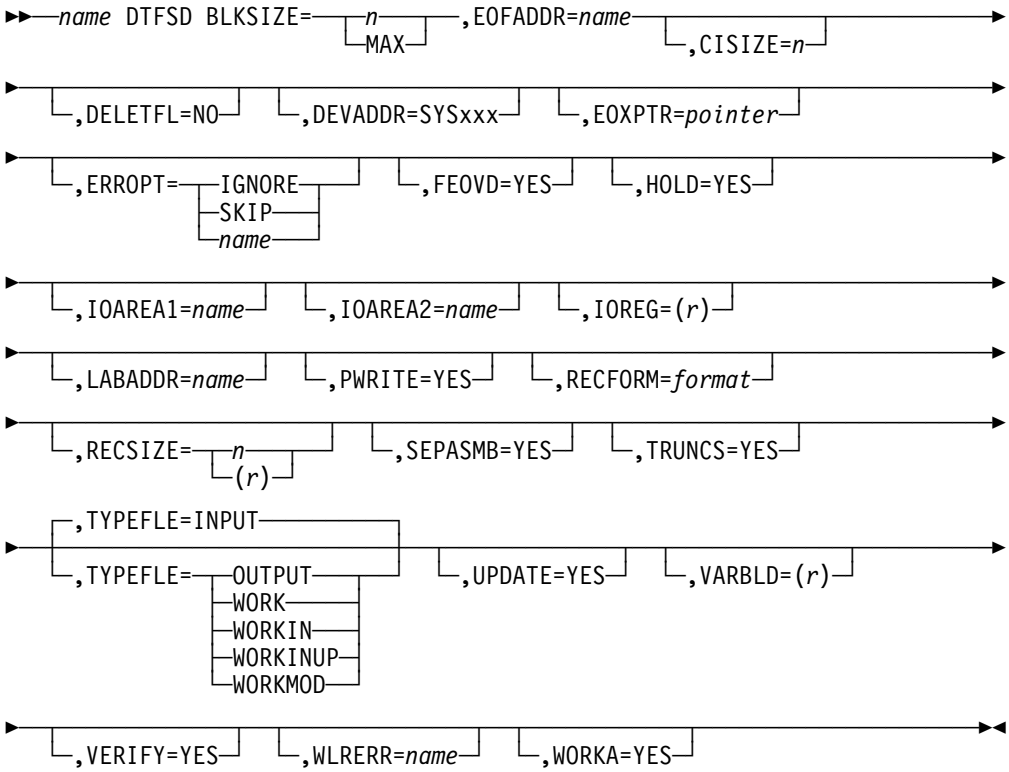
For a printer with the universal character set feature, or for a 3800, this operand determines whether data checks occurring in case of unprintable characters are indicated to the operator or printed as blanks. The operand is especially useful if you are using first-character forms control and have modules that cannot process the CNTRL macro. If the operand is omitted, OFF is the default.

ON Data checks are processed with an operator indication.

OFF Data checks are ignored and blanks are printed for the unprintable character.

WORKA=YES
If output records are processed in work areas instead of in the I/O areas, specify this operand. You must set up the work area in storage. The address of the work area, or a general-purpose register which contains the address, must be specified in each PUT macro.

DTFSD (Define the File for Sequential Disk I/O) Macro



Required RMODE: **24**

The macro defines a disk file for sequential (consecutive) processing. Only IBM standard label formats are processed for the file.

Note: To code a DTFSD macro for a SAM file in VSAM managed space, consult the *VSE/VSAM User's Guide and Application Programming*under "VSE/VSAM Support for SAM Files".

BLKSIZE=n | MAX
For n in BLKSIZE=n, code the length of the I/O area. If the record format is variable or undefined, enter the length of the I/O area needed for the largest block of records.

For input files with fixed-length blocked records, BLKSIZE must be an integer multiple of RECSIZE; for output files, eight bytes must be added for IOCS to set up a count field.

If the file is on an FBA device, the operand specifies the logical block size. For an FBA disk, the maximum value is 32761 (the maximum CISIZE value minus 7).

The BLKSIZE value for output files must include eight bytes for a count field to provide compatibility between FBA and CKD disk.

The value given in this operand can be overridden by the BLKSIZE operand of the DLBL job control statement if you define blocked records (RECFORM=xxxBLK). For an output file, the records are blocked according to the size specified by the BLKSIZE operand (from the DLBL statement if it was specified; otherwise from the DTFSD). For an input file, the BLKSIZE value must match the format of the data as this resides on the disk.

To use the DLBL BLKSIZE operand:

- The device must be a CKD device; else, the operand is ignored.
- Partition GETVIS space for a DTF extension and new buffers must be available.
- DTFSD RECFORM=xxxBLK must have been specified.

Specify BLKSIZE=MAX for optimum use of the storage capacity of your disk device. This sets the length of the I/O area to one full track if:

- The file resides on a CKD device.
- The file's records have a format other than control interval (CI).

OPEN obtains the track capacity of your device and sets the length of the I/O area accordingly.

If the records of the file have the CI format, BLKSIZE=MAX sets the I/O area to a length of 32761 bytes, except when you specify also RECFORM=FIXBLK. In that case, BLKSIZE=MAX sets the length of the I/O area to the highest integer multiple of RECSIZE that is not greater than 32761.

If you specify BLKSIZE=MAX, any CISIZE specification is ignored.

CISIZE=n

This operand specifies the size of the file's control intervals. The operand applies if the file resides on an FBA disk assigned to a non-system file logical unit. The operand is ignored if the device is assigned to a system file (SYSRDR, SYSIPT, SYSLST, or SYSPCH), or to a CKD-disk extent.

The value n must be a multiple of the FBA block size and, if greater than 8K, must be a multiple of 2K. The maximum value is 32768.

If CISIZE is omitted, CISIZE=0 is assumed.

For an output file, the specified control interval size may be overridden when the program is to be executed. You do this by specifying the CISIZE operand of the DLBL control statement (except when you specify BLKSIZE=MAX in this macro).

For an input file, the size stored in the format-1 label is used. If this value is zero, OPEN calculates a value based on your specification for BLKSIZE.

Omit CISIZE if you specify BLKSIZE=MAX.

DELETFL=NO

Specify this operand if the CLOSE macro is not to delete the format-1 and format-3 label for a work file. The operand applies to work files only.

DEVADDR=SYSxxx
This operand must specify the symbolic unit associated with the file if an extent is not provided. A job control EXTENT statement is not required for single-volume input files. If an EXTENT statement is provided, its specification overrides any DEVADDR specification. SYSnnn represents an actual I/O address, and is used in the ASSGN job control statement to assign the actual I/O device address to this file.

EOFADDR=name
This operand specifies the name of your end-of-file routine (for input or work files). IOCS automatically branches to this routine on an end-of-file condition. In this routine, you can perform any operations required at end of file (you generally issue the CLOSE macro). Note that the routine always gets control in 24-bit addressing mode.

EOXPTR=pointer
This operand points to a 4-byte field that contains the address of an end-of-extent exit routine. IOCS branches to this routine when the end of the last (or only) extent is reached during an output operation on an output or work file. Note that the routine always gets control in 24-bit addressing mode.

On entry to the routine, register 15 contains:

0 If the end-of-extent condition occurred during normal processing. In this case, you can issue a CLOSE macro for the file; usually, there will be enough space for an end-of-file record. However, for blocked files, the last block may not be written.

4 If the condition occurred during CLOSE processing. In this case, you can no longer issue an imperative macro for the file. For blocked files, the last block may not be written.

8 If the condition occurred during processing of a POINT macro for a work file (if, for example, the NOTE information provided was incorrect).

EOXPTR is not allowed for TYPEFLE=WORKIN.

ERROPT=IGNORE | SKIP | name
This operand is specified if a job is not to be terminated when a read or write error cannot be corrected in the disk error routines. The disk error routines normally retry failing I/O operations several times before considering the error irrecoverable. Once the error is considered irrecoverable, the job is terminated unless the ERROPT operand is specified.

Note that a no-record-found condition is *not* considered a real I/O error. Therefore the ERROPT exit will not be activated by a no-record-found condition. Instead, the operator receives a message to which he can respond with CANCEL; then the error is considered irrecoverable.

The functions you can specify are explained below:

ERROPT=IGNORE
The error condition is ignored. The records are made available for processing. When reading spanned records, the whole spanned record

or block of spanned records is returned, rather than just the one physical record in which the error occurred.

On output, the physical record or control interval in which the error occurred is ignored as if it were written correctly. If possible, any remaining spanned record segments are written.

ERROPT=SKIP
On input, no records in the error block or control interval are made available for processing. The next block or control interval is read from the disk, and processing continues with the first record of that block. When reading spanned records, the whole spanned record or block of spanned records is skipped, rather than just one physical record.

On output or for an UPDATE=YES file, the physical record or control interval in which the error occurred is ignored as if it were written correctly. If possible, any remaining spanned record segments are written.

ERROPT=name
IOCS branches to the error routine named in this operand. In this routine, you can process or make note of the error condition as desired, but you should not issue any imperative macro instructions for the file invoking the error exit. Note that the routine always gets control in 24-bit addressing mode.

To continue processing at the end of the routine, return to IOCS by either:

- Branching to the address in register 14, or
- Coding the ERET macro.

FEOVD=YES
This operand is specified if a forced end of volume for disk feature is desired. It forces the end-of-volume condition before physical end of volume occurs. When the FEOVD macro is issued, the current volume is closed, and I/O processing continues on the next volume. This operand does not apply to work files.

HOLD=YES
This operand may be specified only if:

1. Generation of the track-hold function was requested for the assembly of your supervisor.
2. Your DTFSD macro includes the operand UPDATE=YES.

For a more detailed discussion of the track-hold function, see “DASD Record Protection (Track Hold)” in the *VSE/ESA System Macro User's Guide*.

IOAREA1=name
This operand specifies, for an input or output file, the symbolic name of the I/O area used by the file. It is not required if WORKA=YES or IOREG=(r) is specified for any input or output file.

If both IOAREA1=name and WORKA=YES are specified on an FBA file, IOAREA1 is ignored.

If the BLKSIZE is overridden by the DLBL statement, and the value is greater than the value specified in the DTF, OPEN issues a GETVIS for the space of the larger I/O area and the specified one is not used.

For variable-length or undefined records, this area must be large enough to contain the largest block or record.

Note: Either IOAREA1=name or WORKA=YES must be specified if variable-length records are to be used with VSE/VSAM managed space.

IOAREA2=name

If two I/O areas are used by GET or PUT, this operand is specified. When variable length records are processed, the size of the I/O area must include four bytes for the block size. For output files, the I/O area must include eight bytes. This operand is ignored if IOAREA1 is not specified.

IOREG=(r)

This operand specifies, for an input or output file, the general purpose register (any of 2 to 12) in which IOCS puts the address of the logical record that is available for processing. At OPEN time, for output files, IOCS puts into the register specified the address of the area where you can build a record. The same register may be used for two or more files in the same program, if desired. If this is done, the program must store the address supplied by IOCS for each record.

This operand must be specified if

- No I/O area has been specified, or
- Blocked input or output records are processed in one I/O area, or
- Two I/O areas are used and the records are processed in both I/O areas.

For an FBA file, the register specified by IOREG will point directly to data in the control interval buffer.

LABADDR=name

Specifies, for an input or output file, the name of the routine in which you process user-standard labels. Note that the routine always gets control in 24-bit addressing mode. For more information about the handling of user-standard labels, see the sections “Processing of User Labels” on page 439.

PWRITE=YES

This operand is specified if formatting output operations to an FBA device (PUT for data files or WRITE SQ for work files) are to cause a physical write for each logical block. If omitted, the actual write takes place only when the control interval buffer is full.

If PWRITE=YES is specified, the POINTR, POINTS, and POINTW macros may not be used.

RECFORM=format

This operand specifies the type of records for input or output. For format, specify one of the following:

FIXUNB For fixed-length unblocked records.
FIXBLK For fixed-length blocked records.
VARUNB For variable-length unblocked records.

VARBLK For variable-length blocked records.
SPNUNB For spanned variable-length unblocked records.
SPNBLK For spanned variable-length blocked records.
UNDEF For undefined records.

If RECFORM=SPNUNB or RECFORM=SPNBLK is specified and RECSIZE=(r) is not specified, an assembler diagnostic (MNOTE) is issued, and register 2 is assumed. If WORKA=YES is omitted, an MNOTE is issued and WORKA=YES is assumed. If RECFORM is omitted, FIXUNB is assumed.

If RECFORM=xxxBLK is specified and if the actual device is a CKD device, you can override the BLKSIZE value with the BLKSIZE operand on the DLBL statement at execution time.

For work files, use FIXUNB or UNDEF only.

RECSIZE=n | (r)
Specifies the number of characters in each record either directly or in a register (any one of 2 to 12, where R2 is the default). When and how to use RECSIZE largely depends on the record format:

- RECSIZE is required in direct format for RECFORM=FIXBLK.
- RECSIZE must not be used for RECFORM=FIXUNB | VARUNB | VARBLK. For these file types, the record size is derived from the block size and set to BLKSIZE minus 8 for output files, and BLKSIZE for others.
- RECSIZE is required with register notation for RECFORM=SPNUNB|SPNBLK|UNDEF. In these cases, RECSIZE
  - is required for output files; the length of each record must be loaded into the designated register before issuing a PUT macro.
  - is optional for input files; IOCS returns the length of the record transferred to virtual storage in the designated register.
- RECSIZE must not be specified for work files.

SEPASMB=YES
Include this operand only if the DTFSD macro is to be assembled separately. This produces an object module ready to be cataloged into a suitable sublibrary by the name you used as file name. The name is used as the module's transfer address. If you omit the operand, the assembler assumes that the DTFSD macro is assembled together with your program.

TRUNCS=YES
This operand is specified if FIXBLK disk files contain short blocks embedded within an input file or if the input file was created with a module that specified TRUNCS. This entry is also specified if the TRUNC macro is issued for a FIXBLK output file. The operand does not apply to work files.

TYPEFLE=INPUT | OUTPUT | WORK | WORKIN | WORKINUP | WORKMOD
Use this operand to indicate whether the file is an input or an output or a work file.

INPUT The GET macro must be used.
OUTPUT The PUT macro must be used.

WORK The READ and WRITE, NOTE and POINTx, and CHECK macros must be used, and RECFORM must be either FIXUNB or UNDEF.

WORKIN Indicates an input type OPEN for which WRITE is not allowed. See Note below.

WORKINUP Indicates an input type OPEN for which WRITE is allowed. See Note below.

WORKMOD Equal to TYPEFLE=WORKINUP if the file already exists. Equal to TYPEFLE=WORK if the file does not exist.

Note: For work files all extents must reside on a single volume. If WORKIN or WORKINUP is specified, the work file has to reside on one volume, that is, only the first EXTENT statement (if any) is processed; additional EXTENT statements are ignored. From the first EXTENT statement, only the logical unit and volume serial number (if specified) are used. Any other information in the EXTENT statement is ignored.

If the operand is omitted, INPUT is assumed.

UPDATE=YES

This operand must be included if the disk input file is updated - that is, if disk records are read, processed, and then re-written in the same disk record locations from which they were read. CLOSE writes any remaining records in sequence onto the disk. For workfiles to be updated, use TYPEFLE= WORKINUP.

This operand is invalid for a file on a disk assigned to a system logical unit (SYSRDR, SYSIPT, SYSLST, or SYSPCH). If a PUT is attempted to an input file, the job will be terminated.

VARBLD=(r)

Whenever variable-length blocked records are built directly in the output area (no work area specified), this entry must be included. It specifies the number (r) of a general-purpose register (any one of 2 to 12), which will always contain the length of the available space remaining in the output area.

IOCS calculates the space still available in the output area, and supplies it to you in the designated register after the PUT macro is issued for a variable-length record. You then compare the length of your next variable-length record with the available space to determine if the record fits in the area. This check must be made before the record is built. If the record does not fit, issue a TRUNC macro to transfer the completed block of records to the file. Then, the present record is built at the beginning of the output area in the next block.

VERIFY=YES

This operand is included if you want to check the parity of disk records after they are written. If this operand is omitted, any records written on a disk are not verified.

WLRERR=name

This operand applies only to disk input files. It does not apply to undefined records. WLRERR specifies the symbolic name of your routine to receive

control if a wrong-length record is read. Note that the routine always gets control in 24-bit addressing mode.

If the WLRERR operand is omitted but a wrong-length record is detected by IOCS, one of the following conditions results:

- If the ERROPT entry is included for this file, the wrong-length record is treated as an error block and handled according to your specifications for an error (IGNORE, SKIP, or name of error routine).
- If the ERROPT entry is not included, the error is ignored.
- If WORKA=YES is specified, do not destroy R0 when return by ERET IGNORE is set, as R0 may be used as work area address.

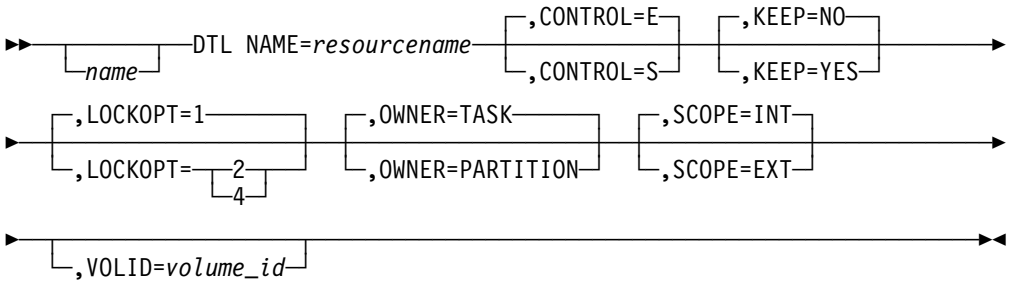
Undefined records are not checked for incorrect record length. The record is truncated when the BLKSIZE specification is exceeded.

WORKA=YES

If records of an input or output file are processed or built in work areas instead of I/O areas, specify this operand. You must set up the work area in storage. The address of the work area, or a general-purpose register which contains the address, must be specified in each GET or PUT macro. For a GET or PUT macro, IOCS moves the record to, or from, the specified work area. WORKA=YES is required for SPNUNB and SPNBLK records, where the work area must be sufficiently long to hold the longest spanned record.

When WORKA=YES is specified, IOREG must be omitted.

DTL (Define the Lock) Macro



Required RMODE: **24**

The macro generates a control block which is used by the LOCK/UNLOCK macros to queue and dequeue a resource access request. The control block, commonly called 'DTL', is generated at the time of program assembly.

NAME=resourcename
Specifies the name by which the resource is known to the system for the purpose of access share control. It is by this name that the system controls shared access of the resource as requested by active tasks via the LOCK macro. These tasks may all be active in one partition, or they may be distributed over several partitions; the resource-share control extends across partitions.

The name may be up to twelve bytes long. If it is shorter, it is padded with blanks. Note that the name must not begin with any of the characters A through I or V, because these characters are reserved for IBM usage.

CONTROL=E | S
Defines how the named resource can be shared while your program owns it, which is determined by this specification and your specification for the operand LOCKOPT. A specification of E means the resource is queued for exclusive use; a specification of S means the resource is queued as sharable.

KEEP=NO | YES
This operand may be used to lock the named resource beyond job step boundaries. Only a main task should use this operand. KEEP=NO indicates that the named resource once locked, is to be released automatically at the end of the particular job step. With KEEP=YES, a named resource that is locked remains locked across job steps; it will be automatically released at end of job.

If a job terminates abnormally, all resources with KEEP=YES are unlocked by the abnormal termination routine.

LOCKOPT=1 | 2 | 4
This operand, together with the CONTROL operand, determines how the system controls shared access in response to a LOCK request.

LOCKOPT=1 and CONTROL=E
No other task is allowed to use the resource concurrently.

LOCKOPT=1 and CONTROL=S
Other 'S' users are allowed concurrent access, but no concurrent 'E' user is allowed.

LOCKOPT=2 and CONTROL=E
No other 'E' user gets concurrent access; however, other 'S' users can have access to the resource.

LOCKOPT=2 and CONTROL=S
Other 'S' users can have concurrent access and, in addition, one 'E' user is allowed.

LOCKOPT=4 and CONTROL=E
No other 'E' user **from another system** is allowed concurrent access. However, other 'S' users from other systems may use the resource concurrently. (Within his own system, the user always has access to the resource.)

LOCKOPT=4 and CONTROL=S
Other 'S' users can have concurrent access and, in addition, one 'E' user **from another system** is allowed.

Note: If the DASDSHR support is not generated in the supervisor, the LOCK request for the resource is always granted.

All users of a particular resource have to use the same LOCKOPT specification when they lock the resource. (Exception: if LOCKOPT=1 and CONTROL=E, the lock status may be modified.)

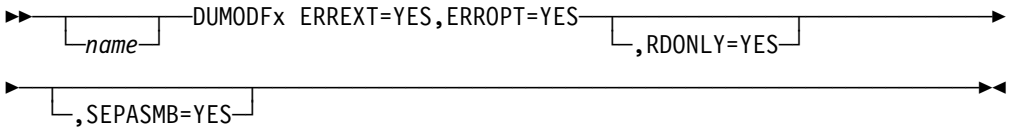
OWNER=TASK | PARTITION
Defines whether the named resource, once locked, can be unlocked only by the task which issued the corresponding LOCK request (OWNER=TASK), or whether it can be unlocked by any task within the partition (OWNER=PARTITION).

When OWNER is defined as PARTITION, a LOCK request for the resource must not specify FAIL=WAIT, FAIL=WAITC, or FAIL=WAITECB, because deadlock prevention (return code 16) is not supported with OWNER=PARTITION.

SCOPE=INT | EXT
This operand may be used for locking resources across systems. SCOPE=EXT specifies that the lock is used across systems. You may omit the operand if you want to lock your resources only on one system since the default is SCOPE=INT (that is, the locking applies to one system only).

VOLID=volume-id
Specifies the 6-byte identifier of a disk volume which (at the time of the LOCK request) is to be checked whether it is mounted on an I/O device that is defined as being shared across systems. If the device is a shared disk, the LOCK request is treated as being defined with SCOPE=EXT; otherwise, SCOPE=INT is assumed, and a SCOPE=EXT specification is ignored.

DUMODFx (Diskette Unit I/O Module Definition) Macro



Requirements for the caller:

AMODE: 24

RMODE: 24

ASC Mode: Primary

The macro defines a logic module for a diskette file. Use either of the following:

DUMODFI – Diskette Unit MODule, Fixed length records, Input file.

DUMODFO – Diskette Unit MODule, Fixed length records, Output file.

ERREXT=YES
Include this operand if permanent errors are returned to a problem program ERROPT routine or if the ERET macro is used with the DTF and module. The ERROPT operand must be specified for this module.

ERROPT=YES
This operand applies to both DUMODFx macros. This operand is included if the module handles any of the error options for an error chain. Logic is generated to handle any of the three options (IGNORE, SKIP, or name) regardless of which option is specified in the DTF. This module also processes any DTF in which the ERROPT operand is not specified.

If this operand is not included, your program is canceled whenever a permanent error is encountered.

RDONLY=YES
This operand causes a read-only module to be generated. If this operand is specified, any DTF used with this module must have the same operand.

SEPASMB=YES
Include this operand only if the module is to be assembled separately. This produces an object module ready to be cataloged into a suitable sublibrary, either by the standard name or by the user-specified name. The name is used as the module's transfer address. If you omit the operand, the assembler assumes that the DUMODFx macro is assembled together with the DTF in your program.

Standard DUMOD Names

Each name begins with a 4-character prefix (IJND) and continues with a 4-character field corresponding to the options permitted in the generation of the module, as shown below. DUMODFx name = IJNDabcd

Char.	Content	Specified Option
a	I	DUMODFI (as the macro operation code)
	O	DUMODFO (as the macro operation code)

Char.	Content	Specified Option
b	C	ERROPT=YES and ERREXT=YES
	E	ERROPT=YES
	Z	Neither is specified
c	Z	Always
d	Y	RDONLY=YES
	Z	RDONLY is not specified

Subset/Superset DUMOD Names

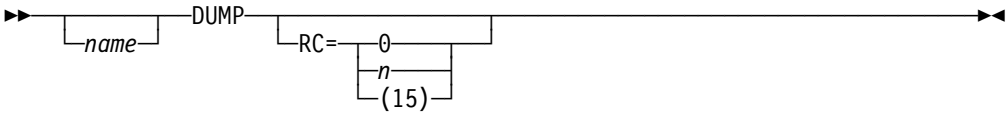
The following chart shows the subsetting and supersetting allowed for DUMOD names.

```
                *  +  *  *
   I  J  N  D   I  C  Z  Y
                0  E     Z
                   Z
```

+ Subsetting/supersetting permitted

***** No subsetting/supersetting permitted

DUMP (Dump Request) Macro



Requirements for the caller:

AMODE: 24 or 31

RMODE: 24

ASC Mode: Primary

This macro provides a hexadecimal dump of the following:

- The contents of the entire supervisor area and the system GETVIS area, or of some supervisor control blocks only.
- The contents of the partition that issued the macro.
- The contents of the registers.

The dump includes the contents of just some of the supervisor control blocks (rather than the entire supervisor area) if either is true:

- The STDOPT job control command specifies DUMP=PART or DUMP=NO.
- A job control // OPTION statement with PARTDUMP or NODUMP is submitted.

The macro causes the job step to be terminated if DUMP was issued by the main (or only) task of the program. If DUMP was issued by a subtask, the macro causes that subtask to be detached without terminating the main task in the partition.

If the job control option SYSDUMP is active, the output of the dump is directed to the dump sublibrary of the partition. If NOSYSDMP is active, the output is directed to SYSLST. If SYSLST is assigned to tape, this tape must be positioned as desired.

If SYSLST is assigned to an IBM 3211 and indexing was used before you issue the DUMP macro, a certain number of characters on every line of the printed dump may be lost. To avoid this, reload the printer's FCB (forms control buffer) by issuing an LFCB macro before you issue the DUMP macro. The FCB image you load must not have an indexing byte.

If DUMP is issued by a job running in real mode, the storage contents of the partition are dumped only up to the limit as determined by the SIZE operand of the EXEC job control statement, plus the storage obtained dynamically through the GETVIS macro. If SIZE was not specified, the entire partition will be dumped. If DUMP is issued by a program running in virtual mode, the entire partition is dumped.

RC=0 | n | (15)
Indicates a user-specified return code, between 0 and 4095, which is passed to job control to reflect the result of the job step and to allow conditional execution of subsequent job steps.

If register notation is used, the return code must be contained in bytes 2 and 3 of the register.

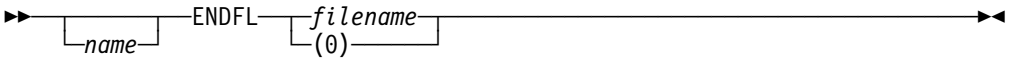
If the operand is omitted, no return code is passed to job control.

The RC operand is only applicable for the main task; it is ignored if the DUMP macro is issued by a subtask.

Note: The DUMP macro modifies some of the user registers:

- If the RC operand is not specified, the macro modifies register 1; all other registers are displayed unchanged.
- If the RC operand is specified, the macro modifies registers 0, 1, 14, and 15; the other registers are displayed unchanged.

ENDFL (End File Load Mode) Macro



Requirements for the caller:

AMODE: 24

RMODE: 24

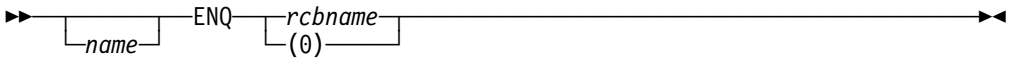
ASC Mode: Primary

The macro ends the load mode started by the SETFL macro. The macro must be issued only after a SETFL and before a CLOSE.

The ENDFL macro performs an operation similar to CLOSE for a blocked file. It writes the last block of data records, if necessary, and then writes an end-of-file record after the last data record. Also, it writes any index entries that are needed followed by dummy index entries for the unused portion of the prime data extent.

filename | (0)
The name of the file to be loaded is the only operand required, and it is the same as the name specified in the DTFIS header entry for the file. The filename can be specified either as a symbol or in register notation. Register notation is necessary if your program is to be self-relocating.

ENQ (Enqueue a Task) Macro



Requirements for the caller:

AMODE: 24 or 31

RMODE: 24 or ANY

ASC Mode: Primary

A task protects a resource by issuing an ENQ (enqueue) macro. When the ENQ macro is issued, the task requesting the resource either continues its execution, or if the requested resource is held by another task, is placed in a wait condition. When the task holding that resource completes, that task issues the DEQ (dequeue) macro. All other tasks that are then waiting for the dequeued resource are freed from their wait condition, and the highest-priority task either obtains or maintains control.

If a task is terminated without dequeuing its queued resources, any task subsequently trying to queue that resource is abnormally terminated. If a task issues two ENQs without an intervening DEQ for the same resource, the task is canceled. Also, any task that does not control a resource but attempts to dequeue that resource is terminated, unless DEQ appears in the abnormal termination routine, in which case it is ignored.

Although the main task does not require the program to set up a task-to-task communication ECB to queue and dequeue, every subtask using the facility must have the ECB operand in the ATTACH macro. That ECB must not be used for any other purpose. Also, a resource can be protected only within the partition containing the ECB.

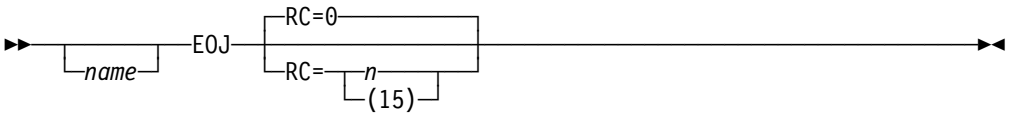
Note: Do not use the ENQ macro in your AB exit routine, since a deadlock may occur.

The ENQ macro supports the 31 bit environment. ENQ may be issued in 24-bit or 31-bit addressing mode, above or below the 16MB line.

When ENQ is issued in AMODE 24, the RCB address is treated as a 24-bit address. When ENQ is issued in AMODE 31, the RCB address is treated as a 31-bit address.

rcbname | 0
Specifies the address of the RCB.

EOJ (End of Job) Macro



Requirements for the caller:

AMODE: 24 or 31

RMODE: 24 or (if RC is not specified) ANY

ASC Mode: Primary

Issue the macro in the main (or only) task within a partition, to inform the system that the job step is finished. If a subtask issues an EOJ, the subtask is detached and the remainder of the partition continues. If the main task issues EOJ, any abnormal termination exits set up via STXIT AB are taken for the subtasks still attached.

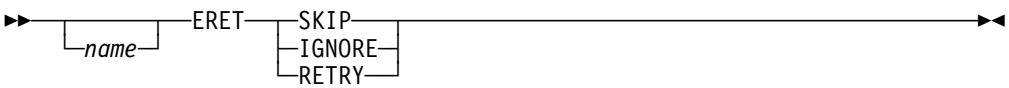
RC=0 | n | (15)
Indicates a user-specified return code, between 0 and 4095, which is passed to job control to reflect the result of the job step and to allow conditional execution of subsequent job steps.

If register notation is used, the return code must be contained in bytes 2 and 3 of the register.

Note that if this operand is included, execution of the EOJ macro is allowed below the 16MB line only (RMODE 24).

If the operand is omitted, a return code of 0 is assumed.

ERET (Error-Handling Return) Macro



Requirements for the caller:

AMODE: 24

RMODE: 24

ASC Mode: Primary

This macro enables your program's ERROPT or WLRERR routine to return to IOCS and specify an action to be taken. The macro applies to:

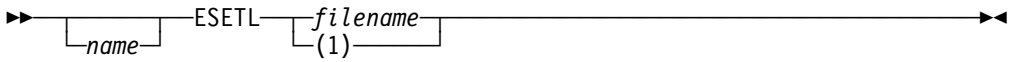
- DTFSD files.
- DTFIS and DTFDU files if ERREXT=YES is specified in the DTFxx macro.
- DTFMT input files if ERREXT=YES is specified.

SKIP Passes control back to the logic module to skip the block of records or control interval in error and process the next one. For disk or diskette output, an ERET SKIP is treated as an ERET IGNORE.

IGNORE
Passes control back to the module to ignore the error and continue processing.

RETRY
Causes the module to retry the operation that resulted in the error. For SD wrong-length record errors, RETRY cancels the job.

ESETL (End Set Limit) Macro



Requirements for the caller:

AMODE: 24

RMODE: 24

ASC Mode: Primary

The macro ends the sequential mode initiated by the SETL macro. If the records are blocked, ESETL writes the last block back if a PUT was issued.

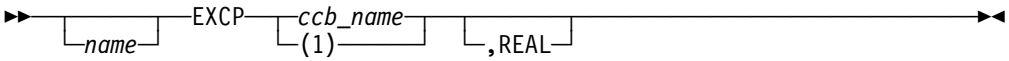
If ADDRTR and/or RANSEQ are specified in the same DTF, ESETL should be issued before issuing a READ or WRITE.

Another SETL can be issued to restart sequential retrieval.

Sequential processing must always be terminated by issuing an ESETL macro.

filename | (1)
Is the same name as the name specified in the DTFIS header entry. The name can be specified as a symbol or in register notation. Register notation is necessary if your program is to be self-relocating.

EXCP (Execute Channel Program) Macro



Requirements for the caller:

AMODE: 24 or 31

RMODE: 24 or ANY

ASC Mode: Primary

The macro requests physical IOCS to start an input/output operation for a particular I/O device.

Physical IOCS determines the device from the CCB or IORB control block specified by blockname. Physical IOCS places the block in a queue and returns control to the problem program. Physical IOCS causes the channel program to be executed as soon as the channel and device are available. I/O interruptions are used to process I/O completion and to start I/O requests if the channel or device was busy at the time the EXCP was executed.

ccbname | (1)
Is the virtual address of the control block established for the device. It can be given as a symbol or in register notation.

The address of the CCB or IORB must be a 24-bit address.

REAL Indicates that the addresses in the CCWs and the address in the control block pointing to the first CCW have already been translated into real addresses. The operand causes the CCW translation routine to be skipped. (For a program running in real mode, the operand is ignored.)

In your program, the EXCP macro with the REAL operand must be preceded by the PFIX macro that causes the system to:

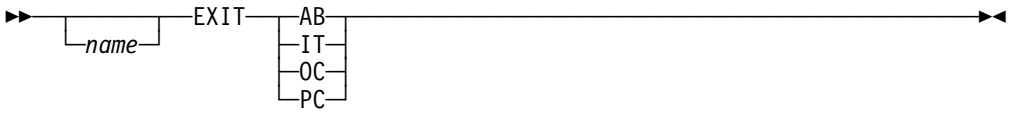
- Page in those program pages that contain the associated control block, channel program, I/O areas, and IDA (indirect address) words (if used).
- Fix these pages in storage.

Notes:

1. With option REAL, if the I/O area being used crosses page boundaries, the data address in your CCW(s) must point to the required indirect data address words within your program. In addition, bit 37 (the IDA bit) of these CCWs must be set to 1. If REAL is not specified, the IDA bit must be set to 0.
2. A channel program has to start with:
   - A long seek command in the case of a CKD disk.
   - A define extent command in the case of an FBA disk.

   The data chaining must be set to zero for these commands.

EXIT (Return from Exit Routine) Macro



Requirements for the caller:

AMODE: 24 or 31

RMODE: 24 or ANY

ASC Mode: Primary

The macro is used to return control from your exit routine to normal processing. Your routine is specified in the STXIT macro. The operands have the following meanings:

AB Exit from your abnormal task termination routine of your main task.

IT Exit from your interval timer routine.

OC Exit from your routine which handles the operator attention interrupt of your main task.

PC Exit from your program check routine.

Issue the EXIT macro only in the corresponding (AB, IT, OC, PC) exit routine; a cancel condition (illegal SVC) may occur if this rule is not observed.

The EXIT **AB** macro may be used only in main tasks. In a subtask, it would result in a cancel condition (illegal SVC). The EXIT AB macro should be issued as early as possible to enable recovery from other abnormal termination conditions. EXIT AB does not restore the original state existing prior to entry of the exit routine, that is, it does not return to the interrupted program. Control is returned from the abnormal termination routine to the instruction following the EXIT AB macro. The cancel condition and ABEND indication of the affected task are reset.

When your AB exit routine completes with the EXIT AB macro, the program continues with an empty linkage stack.

For **IT, OC, and PC,** the interrupt status information and registers are restored from the exit save area. (From the extended save area, also the access registers are restored.) If the save area is unchanged, control is returned to the instruction in your interrupted program immediately after the instruction where the interruption occurred. By changing the save area, you can cause your program to continue from a different point or with changed register contents.

When your PC, IT, or OC exit routine completes with the corresponding EXIT PC|IT|OC macro, the program continues with the linkage stack existing at the time the EXIT macro was issued. When the exit routine completes with EXIT PC|IT|OC, the linkage stack must be equal to the linkage stack at exit entry. Otherwise the program is canceled.

An exit routine using non-paired instructions (BAKR|PC, PR) is canceled.

STXIT Macro Issued With AMODE 24

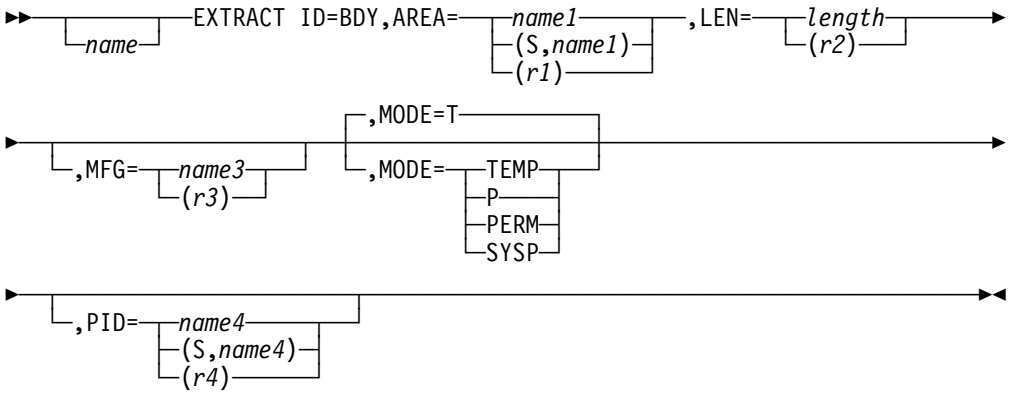
- If the address part of SVUPSW (see “MAPSAVAR (Map Save Area) Macro” on page 309) has not been changed, control is returned to the instruction (and in the same mode) where the interruption occurred.
- If the address part of SVUPSW has been changed, control is returned to the address and with the status that is indicated by the PSW in field SVUPSW. Since this is a BC-mode PSW, neither AR (access register) mode nor a continuation address larger than X'FFFFFF' can be specified.
- If the exit routine uses the address part indicated by the PSW in field SVUPSW for problem analysis, unpredictable results (even a program check) may occur if the interruption takes place in a program part above the 16MB line.

STXIT Macro Issued With AMODE ANY

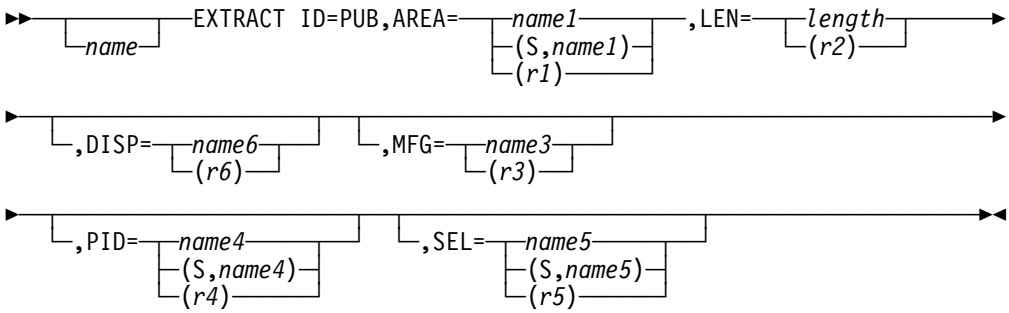
- The program continues processing with the PSW specified at SVUAPSW (bytes 0 and 1 of the PSW are retrieved from the PSW at the time of interruption) and with the general-register and access-register content retrieved from the exit save area.

EXTRACT (Extract Control Information) Macro

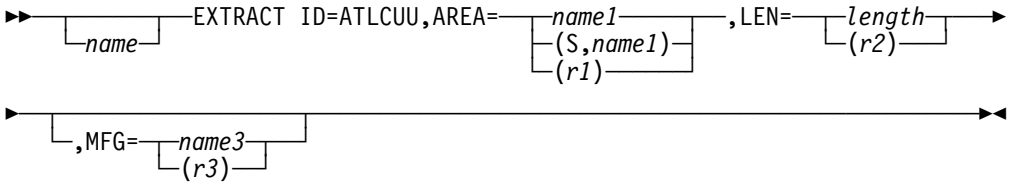
To display partition boundaries:



To display unit information:



To retrieve a free tape drive in an automated tape library:



Requirements for the caller:

AMODE: 24 or 31

RMODE: 24 or ANY

ASC Mode: Primary

The macro may be used to retrieve and display partition boundaries or unit information (from the PUB table). It may also be used to retrieve a free tape drive for a subsequent LBSERV MOUNT request.

The retrieved information can be interpreted with the help of the MAPEXTR (or MAPBDY, MAPBDYVR, MAPSYSP) and IJBPUB macros.

ID=BDY
Indicates that partition boundaries are to be displayed.

ID=PUB
Indicates that unit information is to be displayed.

ID=ATLCUU
Indicates that a free tape drive in an automated tape library is to be retrieved for a subsequent LBSERV MOUNT request.

If the tape library is VM-controlled, VSE searches for a free, not operational tape drive which was added as 3490E or TPA. The specified external device type and logical library name are ignored in this case. If the tape library is not controlled by VM, VSE searches for a free, operational tape drive belonging to the requested library.

In both cases, if a free cuu is found, the mount ownership flag is set and the cuu is returned to the caller. Mount ownership can only be reset by EOJ processing.

AREA=name1 | (S,name1) | (r1)
Specifies the address of the area where the extracted information is to be stored. For ID=ATLCUU, the area also contains input information as described in Figure 39 on page 308.

LEN=length | (r2)
Specifies the length of the area as an integer, a self-defining term, or as a value in a register.

DISP=name6 | (r6)
Defines the offset within the PUB table entry of the specified device. DISP may be specified as a number, a register containing the displacement value, or the field name in the DSECT generated by the IJBPUB macro.

MFG=name3 | (r3)
The MFG operand is required if the program is to be reenterable. It specifies the address of a 64-byte dynamic storage area, that is: storage which your program obtained through a GETVIS macro. The area is required for system use during execution of the macro.

MODE=T | TEMP | P | PERM | SYSP
MODE=T or MODE=TEMP indicates that the temporary boundaries of the issuing partition plus the PFIX limits of the partition are to be returned. Do not specify the PID operand together with MODE=T|TEMP; a snapshot of any other partition's temporary boundaries is unreliable.

If your program runs in real mode, the boundaries of the real partition are returned.

The output can be interpreted with the help of the MAPEXTR macro with the keywords ID=BDY and MODE=TEMP. For compatibility reasons, the MAPBDY macro is still supported.

MODE=P or MODE=PERM indicates that the permanent boundaries (including the PFIX limits) are to be returned of either the issuing partition or of the partition indicated by the PID operand. However, these boundaries correspond to the latest allocation and may not have been used by the job active in the partition.

The output can be interpreted with the help of the MAPEXTR macro with the keywords ID=BDY and MODE=PERM. For compatibility reasons, the MAPBDYVR macro is still supported.

MODE=SYSP indicates that information about the system layout is to be provided, including the values for the 31-bit addressable part of the supervisor. The output can be interpreted with the help of the MAPEXTR macro with the keywords ID=BDY and MODE=SYSP. For compatibility reasons, the MAPSYSP macro is still supported for old programs; however, new fields are no longer mapped.

PID=name4 | (S,name4) | (r4)
Specifies the address of a two-byte field containing the identification key of the partition (PIK) for which the information is retrieved. If this operand is omitted, the identifier of the partition issuing the request is taken as the default.

SEL=name5 | (S,name5) | (r5)
Specifies the address of a halfword containing the logical unit number in the same format as the logical unit number in the CCB.

Return Codes in Register 15

0 The requested information has been extracted and, for ID=ATLCUU, the mount ownership is set.

4 The partition specified is not supported in the system. For ID=ATLCUU, an invalid external device type has been specified.

8 The logical unit specified exceeds the range of the logical units for the specified partition.

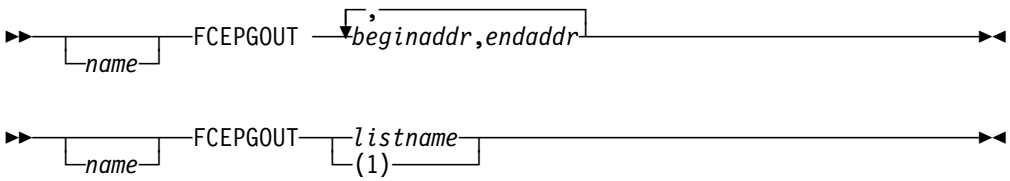
12 The LUB is not assigned or ignored. In addition, an indicator is stored in byte 0 of the area indicated by the AREA operand. This indicator is either

X'FF' if the logical unit is unassigned, or
X'FE' if the logical unit is assigned IGN.

16 The length value is zero, negative, or below the minimum, or the DISP specification exceeds the length of the PUB entry.

20 For ID=ATLCUU, no free device has been found.

FCEPGOUT (Force Page Out) Macro



Requirements for the caller:

AMODE: 24 (if SPLEVEL SET=1)

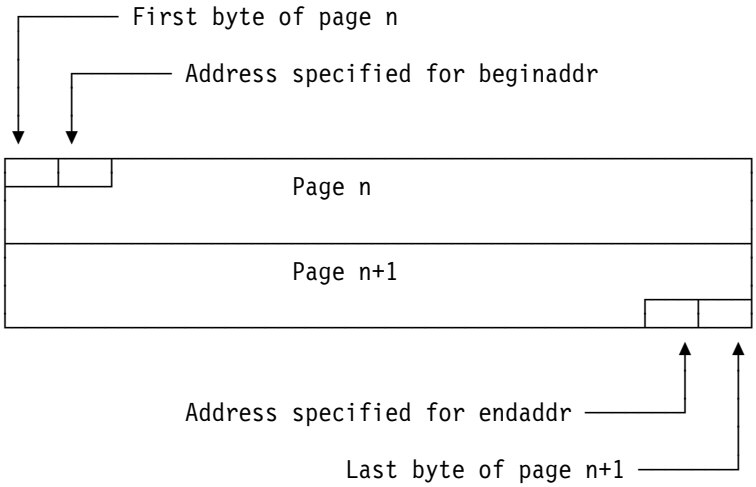
24 or 31 (if SPLEVEL SET>1)

RMODE: 24 (if SPLEVEL SET=1)

24 or ANY (if SPLEVEL SET>1)

ASC Mode: Primary

The macro causes a certain area of virtual storage to be scheduled for page-out if it is in processor (real) storage. This request is ignored if the specified area does not contain a full page. This can happen up to an area size of twice the page size minus two bytes as shown below.



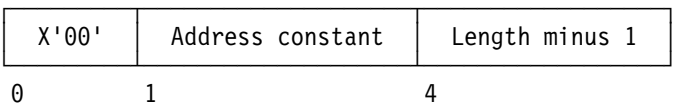
In a system without page data set, execution of the macro results in a null operation; the return code is set to zero.

beginaddr
Points to the first byte of the area to be paged out.

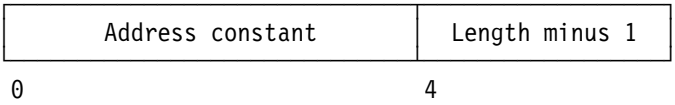
endaddr
Points to the last byte of the area to be paged out.

listname | (1)
Is the name of a list of consecutive 8-byte entries as shown below. (Register notation may also be used.) The address of this parameter list and the addresses in the list are treated as 3-byte addresses if the macro is invoked in 24-bit addressing mode and as 4-byte addresses if invoked in 31-bit addressing mode.

24-bit Addressing Mode:



31-bit Addressing Mode:



where:

Address constant =
Address of the first byte of the area to be paged out.

Length = A binary constant indicating the length of the area to be paged out.

The end of the list is indicated by a non-zero byte following the last entry (for 24-bit mode). For 31-bit mode, a non-zero value in bit 0 of the byte following the last entry indicates the end of the list.

Exceptional Conditions

- The program is running in real mode.
- The page(s) referenced by the macro is (are) outside of the requesting partition.
- A page handling request is active for the referenced page(s).
- The page(s) is (are) not in real storage.
- The page(s) is (are) fixed.

For those pages the FCEPGOUT request will be ignored.

Return Codes in Register 15

0 All specified pages have been forced for page-out or the request has been ignored because the issuing program is running in real mode.

2 The begin address is greater than the end address, or a negative length has been found.

4 At least one of the requested pages does not belong to the partition in which the issuing program is running. The FCEPGOUT request has only been executed for those pages which belong to the partition of the issuing program.

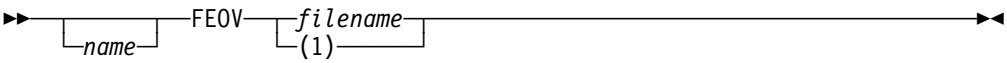
8 At least one of the requested pages is temporarily fixed, either by the system or a previous PFIX macro. The FCEPGOUT request has only been executed for the unfixed pages.

16 The list of areas that are to be paged out is not completely in the requesting program's partition. The request is ignored.

20 Inconsistent function/option code provided in register 15 (not possible if macro interface is used).

Any combination of return codes 2, 4, and 8 is possible.

FEOV (Force End of Volume) Macro



Requirements for the caller:

AMODE: 24

RMODE: 24

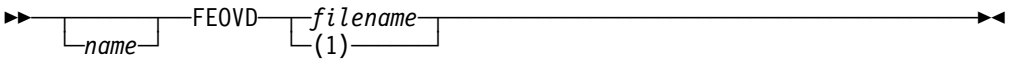
ASC Mode: Primary

The macro is used for files on magnetic tape (programmer logical units only) to force an end-of-volume condition before sensing a reflector mark. This indicates that processing of records on one volume is considered finished, but that more records for the same logical file are to be read from, or written on, a following volume. For system units, see the “SEOV (System End-of-Volume) Macro” on page 364.

When physical IOCS macros are used and DTFPH is specified for standard label processing, FEOV may be issued for output files only. In this case, FEOV writes a tapemark, the standard trailer label, and any user-standard trailer labels if DTFPH LABADDR is specified. When the new volume is mounted and ready for writing, IOCS writes the standard header label and user-standard labels, if any.

filename | (1)
The name of the file is the only operand required. You can specify the name either as a symbol or in register notation.

FEOVD (Force End of Volume for Disk) Macro



Requirements for the caller:

AMODE: 24

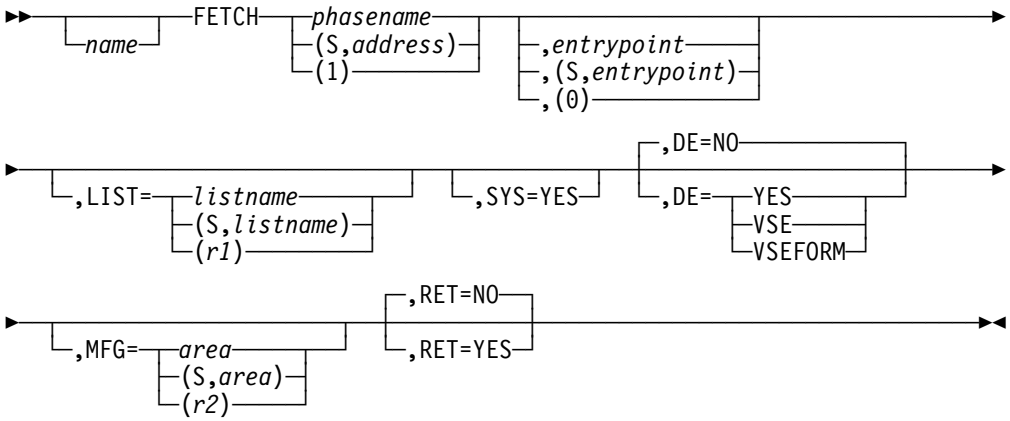
RMODE: 24

ASC Mode: Primary

The macro is used for either input or output files to force an end-of-volume condition before it actually occurs. This indicates that processing of records on one volume is finished, but that more records for the same logical file are to be read from, or written on, the following volume. If extents are not available on the new volume, or if the format-1 label is posted as the last volume of the file, control is passed to the EOF address specified in the DTF.

filename | (1)
The name of the file is the only required operand. The name can be specified either symbolically or in register notation.

FETCH (Fetch a Phase) Macro



Requirements for the caller:

AMODE: 24 or 31

RMODE: 24

ASC Mode: Primary

The macro loads and gives control to the phase specified in the first operand. If the phase is in the SVA, it is not loaded into the partition, but control is given to the phase. For information on how to load phases into the SVA and how to write SVA-eligible (reenterable) phases see “Loading Phases into the SVA” in the *VSE/ESA Guide to System Functions*.

Notes:

1. Both the expanded code and the parameter lists must be below 16MB, otherwise the request will be canceled.
2. The addresses are not validated by the macro expansion. Depending on the macro call, only three bytes of a passed four-byte address may be passed to the SVC service. This may cause unpredictable results, since the address cannot be validated properly.

The AMODE bit in the PSW will be updated depending on the AMODE attribute of the phase:

If AMODE=31, the phase gets control in AMODE 31.
If AMODE=24, the phase gets control in AMODE 24.
If AMODE=ANY, the phase gets control in the AMODE of its caller.

Restriction: The load point plus the size of the phase must be below the start of the GETVIS area.

phasename | (S,address) | (1)

For phasename specify the name of the required phase. The address is regarded as either a 24-bit or 31-bit address, depending on the AMODE of the caller. If the caller has AMODE 31 and the address points to a storage location above 16MB, the requestor is canceled.

If the DE operand is omitted or DE=NO is specified, the address as specified in (S,address) or as loaded into a register points to an 8-byte field that contains the phase name.

For DE=YES or DE=VSE or VSEFORM, the operand has a different meaning; refer to the discussion of the DE operand.

entrypoint | (S,entrypoint) | (0)
Control is passed to the address specified for **entrypoint**. This entry point overwrites the entry point determined a link-edit time. The entry point is regarded either as 24-bit or 31-bit address, depending on the AMODE of the caller, but since the phase must be located totally below 16MB, an entry point above 16MB is regarded as invalid. Apart from that, no further validity checking is done.

If this operand is not specified, control is passed to the entry point determined at link-edit time.

If **entrypoint** is given in register notation, register 1 must not be used. You preload the register with the entry-point address.

With S-type notation, the entry point is derived from base register and displacement, for example (S, offset (reg)). If, instead, a symbolic name is used for **entrypoint**, the macro expansion results in a V-type address constant. The entry point does not have to be identified by an EXTRN statement.

LIST=listname | (S,listname) | (r1)
For listname specify the name of the local directory list generated in the partition by the GENL macro. When this operand is included, the system scans the local directory list for the required phase name before it initiates a search for this phase name in the directories of the accessible sublibraries.

If the phase has been found in the local directory list, general register 0 points to the related directory entry; otherwise, register 0 is set to zero.

The local directory list must be located below 16MB (only three bytes are used in the macro expansion).

If LIST is specified, a specification of YES or VSE (or VSEFORM) for DE= is invalid.

SYS=YES
If this is specified, the system scans the system directory list (SDL) in the SVA and the system sublibrary before any private sublibraries. If the operand is omitted, the SDL and the private sublibraries are searched first.

DE=NO | YES | VSE | VSEFORM
By specifying DE=YES or DE=VSE | VSEFORM, you can generate your own local directory entry for a frequently used phase in order to save a time-consuming library directory search for that phase. A specification of YES or VSE|VSEFORM is invalid if LIST is specified.

DE=NO
Indicates that no local directory entry is to be generated.

DE=YES
Indicates a conventional 38-byte directory entry in the old librarian format (prior to VSE/Advanced Functions Version 2).

DE=VSE | VSEFORM
Indicates a 40-byte directory entry in the new librarian format (starting with VSE/Advanced Functions Version 2). VSE is a short form of VSEFORM.

For DE=YES or DE=VSE, the MAPDNTRY macro can be used to interpret the information returned by the FETCH (or LOAD) macro. Among other information, the local directory entry shows the AMODE/RMODE assigned to the phase. The directory entry must be located below 16MB (see explanation for **phasename**).

The local directory entry is activated by the first FETCH request; all further FETCH requests are executed without any directory search.

If the first operand is written as phasename (instead of S-type or register notation), a directory entry will be generated within the macro expansion. The generated directory entry will contain the phasename in the first 8 bytes.

If you use S-type or register notation for the first operand, you must set aside the 38-byte (or 40-byte) field for the directory entry yourself and point to it via this operand. The directory entry must contain the phase name in the first 8 bytes (left-justified and padded with blanks) and have the following format:

For DE=YES:

```
Byte    Contents
  0     CL8'PHASENAM'
  8     XL3'0'
 11     XL1' '     NO. OF HALFWORDS FOLLOWING
 12     XL26'0'
```

For DE=VSE | VSEFORM:

```
Byte    Contents
  0     CL8'PHASENAM'
  8     XL3'FFFFFF'    ID FOR NEW FORMAT
 11     XL1' '         NO. OF HALFWORDS FOLLOWING
 12     XL28'0'
```

MFG=area | (S,area) | (r2)
The operand is required if the program which issues the FETCH macro is to be reenterable. It specifies the address of a 64-byte dynamic storage area, that is, storage which your program obtained through a GETVIS macro. This area is required for system use during execution of the macro.

RET=NO | YES
By specifying RET=YES you can cause control to be returned to your program in any case (both in normal and error situations). Register 15 contains one of the following return codes:

Return Codes in Register 15

0 FETCH completed successfully.

4 Phase not found. This return code will also be issued if a user directory entry is found and the corresponding phase has already been deleted or re-cataloged.

8 Irrecoverable I/O error during FETCH processing.

12 Invalid library or sublibrary structure detected during FETCH processing.

16 Invalid address range detected during FETCH processing – Either of the following:

Local directory entry outside the partition.
Phase does not fit into the partition.
Entry point of phase is not below 16MB. (In case RET=NO is specified, the program is canceled with 'invalid address').

20 Security violation.

24 Inconsistent user directory state: FETCH found an inconsistency between your program's local directory entry and the corresponding entry in the directory of the related sublibrary. The local directory entry is overwritten by the entry read from the directory of the sublibrary. FETCH checks the following:

Length of phase
Relocation state
Difference between the load point and the partition-start address
Difference between the load point and the entry point

28 Partition is too small (or phase does not fit into the logical transient area).

FREE (Free Disk Area) Macro

Requirements for the caller:

AMODE: 24

RMODE: 24

ASC Mode: Primary

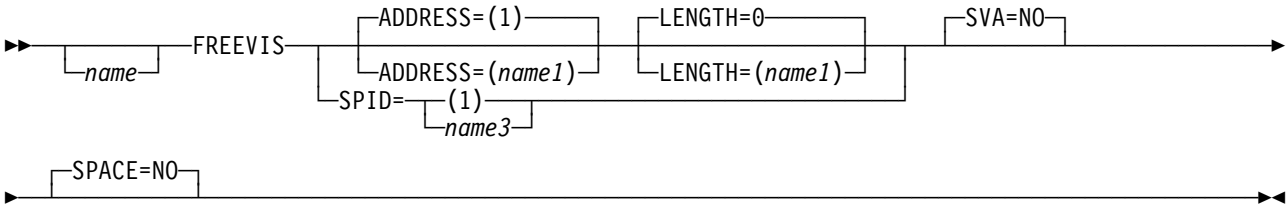
The macro is used in conjunction with the HOLD=YES option of a DTFxx macro. It frees a portion of a disk volume that is being held under disk record (track) protection. On a CKD device, that protected portion is a track; on an FBA device, it is a number of contiguous FBA blocks. On an FBA device, or on a DTFSD or DTFDI file assigned to a CKD disk, the FREE macro is treated as a null operation; all holding and freeing of FBA block ranges or CKD tracks for DTFSD and DTFDI are performed implicitly by LIOCS.

The same track (or blocks) can be held more than once without an intervening FREE if the hold requests are from the same task. The same number of FREEs must be issued before the track (or block) is completely freed. However, a task is terminated if more than 16 hold requests are recorded without an intervening FREE, or if a FREE is issued to a file that does not have a hold request for that track (or block). For situations that require the use of the FREE macro, see “DASD Record Protection (Track Hold)” in the *VSE/ESA System Macro User's Guide*.

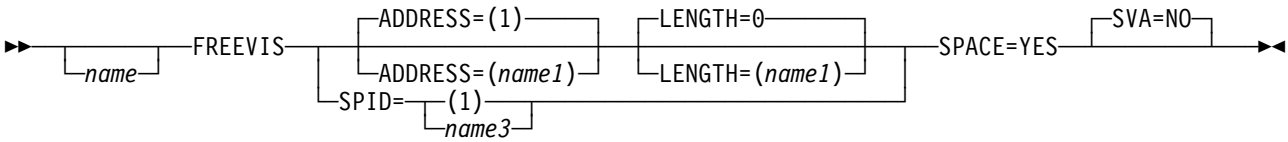
filename | (1)
This operand is the same as the name you specified in the DTFxx macro for the file.

FREEVIS (Free Virtual Storage) Macro

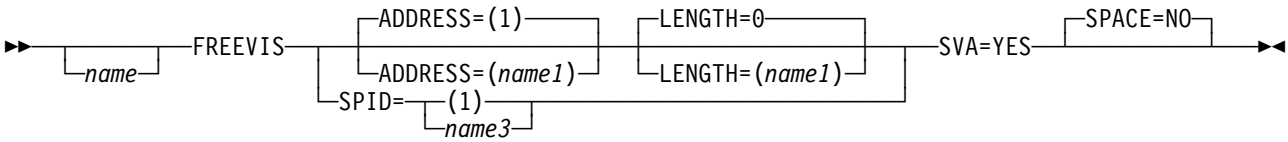
Format 1: Freeing Storage from the Partition FREEVIS Area



Format 2: Freeing Storage from the Space FREEVIS Area



Format 3: Freeing Storage from the System FREEVIS Area



Requirements for the caller:

AMODE: 24 or 31

RMODE: 24 or ANY

ASC Mode: Primary

The macro releases a block of virtual storage that was obtained by the GETVIS macro.

If you code the macro without any operand, the system assumes that the start address of the block to be released is contained in register 1 and that the length of this block was placed into register 0. If the macro is issued without an operand, the macro must not contain a comment unless the comment begins with a comma.

ADDRESS=name1 | (1)
The start address of the virtual storage block to be released in the GETVIS area may be specified either in a 4-byte field addressed by name1 or in a register.

The start address is regarded as a 31-bit address.

LENGTH=name2 | (0)
The length of the virtual storage block to be released may be specified in a 4-byte field addressed by name2 or in a register. The length is specified in number of bytes. The smallest unit of virtual storage that can be released by FREEVIS is either of the following:

128 bytes if the GETVIS area is part of a partition.
16 bytes if the GETVIS area is part of the SVA or of the dynamic space GETVIS area.

If the specified length is not a multiple of 128 or 16, respectively, it is rounded to the next higher multiple of 128 or 16.

SPACE=NO | YES
SPACE=YES indicates that the virtual storage block is to be released from the dynamic space GETVIS area. The requestor needs storage protection key zero.

SPACE=YES must not be specified together with SVA=YES.

SPID=name3 | (1)
Specifies that the whole subpool indicated by the SPID operand is to be freed. The eight-byte subpool ID (addressed either by name3 or by a register) consists of the six-byte subpool name and a two-byte subpool index that was set by the system when the subpool was created via GETVIS.

The address where the SPID operand points to is regarded either as a 24-bit or 31-bit address, depending on the AMODE of the caller.

Note: When the SPID operand is specified, the ADDRESS and LENGTH operands are ignored. Use ADDRESS and LENGTH if only a block of storage within the subpool is to be freed.

SVA=NO | YES
SVA=YES can be specified only in a program that runs with storage protection key zero.

If you specify SVA=YES, the system releases the block in the SVA. If you specify SVA=NO or omit the operand, the system releases the block in the partition in which your program runs.

SVA=YES must not be specified together with SPACE=YES.

Return Codes in Register 15

0 FREEVIS completed successfully.

4 The size of the (real) partition GETVIS area is 0K.

8 The specified length is negative.

12 The specified address is not within the GETVIS area or the address is not a multiple of:

128 bytes if the GETVIS area is part of a partition.
16 bytes if the GETVIS area is part of the SVA or of the dynamic space GETVIS area.

16 The specified storage block to be released (ADDRESS + LENGTH) exceeds the GETVIS area or is not within a subpool.

20 Invalid FREEVIS option.

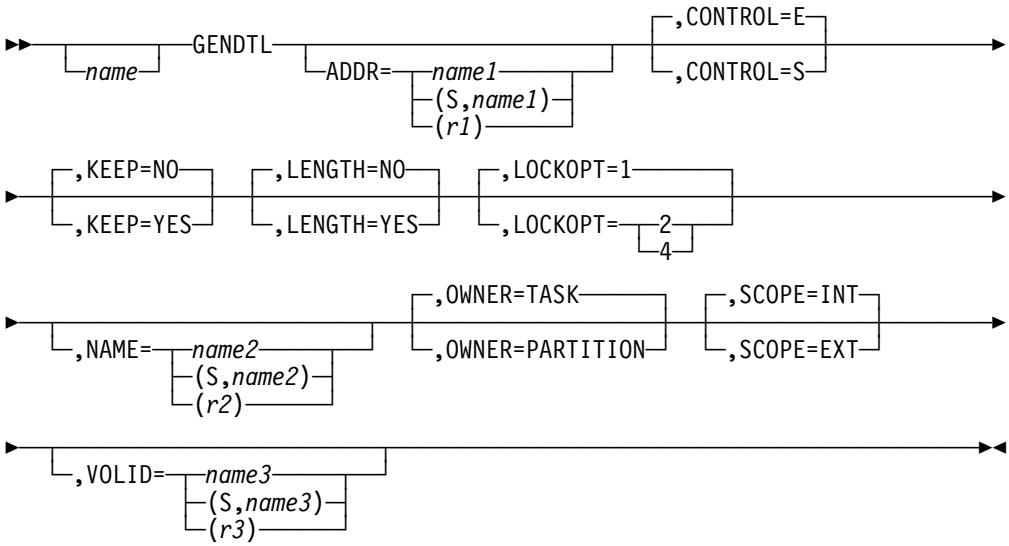
24 An invalid subpool ID was passed.

28 The specified subpool does not exist.

36 An invalid subpool index was specified in the SPID operand. The subpool was created with the GETVIS operand SPCNTRL=YES (compare the GETVIS macro).

40 FREEVIS for an area or subpool for which a PFIX request is pending is not allowed.

GENDTL (Generate the DTL Block) Macro



Requirements for the caller:

AMODE: 24 or 31

RMODE: 24

ASC Mode: Primary

The macro generates a control block which is used by the LOCK/UNLOCK macros to enqueue/dequeue a resource access request. The control block, commonly called 'DTL' (Define The Lock), is generated at the time of program execution. Space for the DTL is either provided by the program, or it is to be acquired by the system.

ADDR=name1 | (S,name1) | (r1)
Specifies the address of the area where the DTL is to be built. If this operand is omitted, storage is allocated in the partition's GETVIS area by an implicit GETVIS request. After a successful GETVIS, the system places the DTL's address in register 1. Register 15 contains the return code set by the implicit GETVIS request.

CONTROL=E | S
Defines how the named resource can be shared while your program owns it, which is determined by this specification and your specification for the operand LOCKOPT. A specification of E means the resource is enqueued for exclusive use; a specification of S means the resource is enqueued as sharable.

KEEP=NO | YES
This operand may be used to lock the named resource beyond job step boundaries. Only a main task should use this operand. KEEP=NO indicates that the named resource once locked, is to be released automatically at the end of the particular job step. With KEEP=YES, a named resource that is locked remains locked across job steps; it will be automatically released at end of job.

If a job terminates abnormally, all resources with KEEP=YES are unlocked by the end-of-task (termination) routine.

LENGTH=NO | YES
If LENGTH=YES is specified, the GENDTL macro returns the length of the DTL in register 0. With LENGTH=NO, register 0 remains unchanged.

LOCKOPT=1 | 2 | 4
Together with the CONTROL operand, this operand determines how the system controls shared access in response to a LOCK request:

LOCKOPT=1 and CONTROL=E
No other task is allowed to use the resource concurrently.

LOCKOPT=1 and CONTROL=S
Other 'S' users are allowed concurrent access, but no concurrent 'E' user is allowed.

LOCKOPT=2 and CONTROL=E
No other 'E' user gets concurrent access; however, other 'S' users can have access to the resource.

LOCKOPT=2 and CONTROL=S
Other 'S' users can have concurrent access and, in addition, one 'E' user is allowed.

LOCKOPT=4 and CONTROL=E
No 'E' user from another system is allowed concurrent access. An 'S' user from another system may use the resource concurrently. Within your own system, you always have access to the resource.

LOCKOPT=4 and CONTROL=S
'S' users can have concurrent access and, in addition, one 'E' user from another system is allowed.

Note: If the DASDSHR support is not generated in the supervisor, the LOCK request for the resource is always granted.

All users of a particular resource have to use the same LOCKOPT specification when they lock the resource. Exception: if LOCKOPT=1 and CONTROL=E, the lock status may be modified.

NAME=name2 | (S,name2) | (r2)
Specifies the address of the area where a 12-byte long resource name is stored. If the name is shorter than 12 bytes, it must be padded with blanks. It is by this name that VSE controls shared access of the resource as requested by active tasks via the LOCK macro. These tasks may all be active in one partition, or they may be distributed over several partitions; the resource-share control extends across partitions.

The name must not begin with any of the characters A through I or V because these characters are reserved for IBM usage.

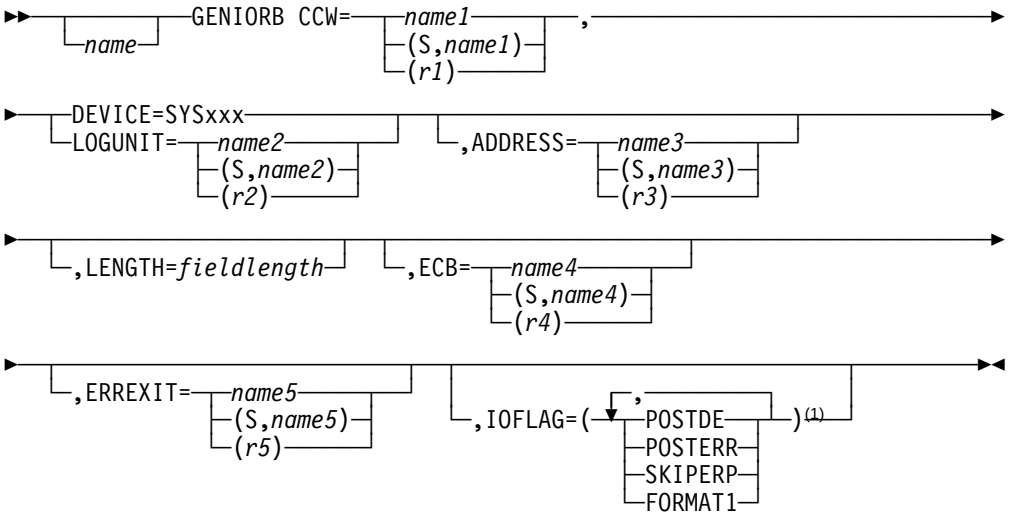
OWNER=TASK | PARTITION
Defines whether the named resource, once locked, can be unlocked only by the task which issued the corresponding LOCK request (OWNER=TASK), or whether it can be unlocked by any task within the partition (OWNER=PARTITION).

When OWNER is defined as PARTITION, a LOCK request for the resource must not specify FAIL=WAIT, FAIL=WAITC, or FAIL=WAITECB, because deadlock prevention (return code 16) is not supported with OWNER=PARTITION.

SCOPE=INT | EXT
This operand may be used for locking resources across systems. SCOPE=EXT specifies that the lock is used across systems. You may omit the operand if you want to lock your resources only on one system since the default is SCOPE=INT (that is, the locking applies to one system only).

VOLID=name3 | (S,name3) | (r3)
Specifies the address of a 6-byte identifier of a DASD volume which (at the time of the LOCK request) is to be checked whether it is mounted on an I/O device that is defined as being shared across systems. If the device is a shared DASD, the LOCK request is treated as being defined with SCOPE=EXT (that is, the SCOPE operand is ignored); otherwise, SCOPE=INT is assumed.

GENIORB (Generate an IORB) Macro



Note:
[1] Each option can be specified once.

Note: The operands FIXLIST and FIXFLAG have become obsolete; for compatibility reasons they are, however, still accepted.

Requirements for the caller:

AMODE: 24

RMODE: 24

ASC Mode: Primary

The macro generates an IORB (input/output request block). The block is generated at the time of program execution. For the layout and contents of an IORB, see Figure 20 on page 224. The IORB is an alternative to the CCB; instead of specifying a CCB in the EXCP macro, the address of an IORB is given.

After execution of the macro:

- register 1 contains the address of the IORB, and
- register 15 contains the return code from an implicit GETVIS.

For a detailed display in your assembly, showing the IORB fields and their meaning, issue the IORB macro with the (only) operand DSECT=YES.

CCW=name1 | (S,name1) | (r1)
This operand gives the name of the first CCW used with the IORB. The name must be the same as the name specified in the assembler CCW statement that builds the CCW.

DEVICE=SYSxxx
This operand specifies the logical unit for the actual I/O unit with which the IORB is associated.

LOGUNIT=name2 | (S,name2) | (r2)
This operand describes the device in logical unit format. It points to a halfword with the same format as a logical unit number (bytes 6 and 7) in a CCB; see Figure 2 on page 57 provided in context with the discussion of the CCB macro.

ADDRESS=name3 | (S,name3) | (r3)
If specified, this operand gives the name of the area in which the IORB is to be generated. If the ADDRESS operand is specified, the LENGTH operand should be specified as well.

Omitting the ADDRESS operand indicates that the required area is to be obtained through an implicit GETVIS issued by the system.

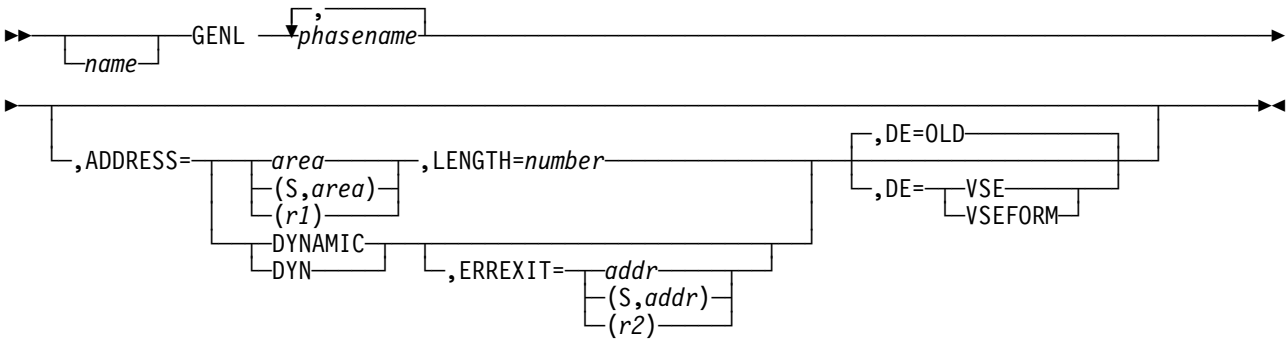
LENGTH=fieldlength
This operand gives the length of the field provided for IORB generation. The value must be given as a self-defining term. If this operand is omitted, a default value equal to the length of the IORB will be used; however, the assembler issues an MNOTE. If the ADDRESS operand is omitted, LENGTH is not used.

ECB=name4 | (S,name4) | (r4)
This operand specifies the address of the ECB to be posted when the I/O is complete. For a more detailed description of an ECB, see the description of the ECB operand of the ATTACH macro on page 26.

ERREXIT=name5 | (S,name5) | (r5)
The operand specifies the address of a routine to be executed should the system be unable to obtain the required virtual storage. If the ERREXIT operand is omitted, failure to obtain virtual storage causes the system to cancel the program (task).

IOFLAG=
For a description of this operand, see the “IORB (I/O Request Block Definition) Macro” on page 224.

GENL (Generate Directory List) Macro



Requirements for the caller:

AMODE: 24 or 31

RMODE: 24

ASC Mode: Primary

The macro generates a local directory list within the partition. This saves access time if you load the same phases more than once in the course of program execution.

The format of the generated directory entries is determined by the DE operand.

phasename1,phasename2,...
Specify, for these operands, the names of phases for which entries in a local directory list are to be built. The list will be generated in alphameric sequence. You may specify up to 200 phase names, but no more than a total of 200 operands.

ADDRESS=area | (S,area) | (r1)
If the ADDRESS operand is omitted, the assembler builds a 38-(or 40-)byte directory entry within the macro expansion for each of the specified phases and inserts the pertinent phase name in the entry. The rest of the entry is filled in by the supervisor when the phase is called by a FETCH or LOAD macro with the LIST option for the first time. When, subsequently, the phase is called again, the entry is active.

Coding ADDRESS in conjunction with LENGTH indicates that the directory is to be built, during execution, at a location within your program whose address is given by the ADDRESS operand.

The specified address must be below 16MB.

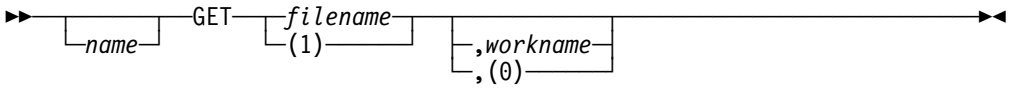
LENGTH=number
LENGTH gives the length of the field provided for the generation of the directory. If LENGTH is too short, the assembler issues an MNOTE.

ADDRESS=DYNAMIC | DYN
Coding ADDRESS=DYNAMIC (or the short form ADDRESS=DYN) directs the system to acquire, through a GETVIS, as much dynamic storage as needed. Execution of the macro causes the contents of registers 0, 1, 14, and 15 to be overwritten.

ERREXIT=addr | (S,addr) | (r2)
ERREXIT is the address of a routine to be executed should the implicit GETVIS fail. If the ERREXIT operand is omitted, an unsuccessful GETVIS will cause the task to be canceled.

DE=OLD | VSE | VSEFORM
Indicates the format of the directory entries to be generated. DE=OLD generates 38-byte directory entries in the old (VSE/Advanced Functions Version 1) format. DE=VSE | VSEFORM generates 40-byte directory entries in the new (Version 2) librarian format. VSE is a short form of VSEFORM.

GET (Get a Record) Macro



Requirements for the caller:

AMODE: 24

RMODE: 24

ASC Mode: Primary

GET makes the next sequential logical record from an input file available for processing in either an input area or in a specified work area. It is used for any input file in the system, and for any type of record.

If GET is used with a file containing checkpoint records, the checkpoint records are bypassed automatically.

filename | (1)
This operand must be the same as the name of the DTF macro for the file from which the record is to be retrieved. The name can be specified as a symbol or in (special or ordinary) register notation (to make your program self-relocating). The high-order bits of the register must be zero, or unpredictable results may occur.

workname | (0)
This operand specifies a work area name or a register (in either special or ordinary register notation) containing the address of the work area. The work area address should never be preloaded into register 1.

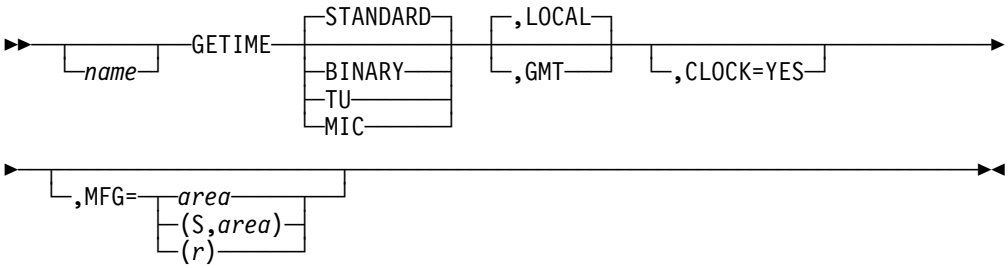
The operand is used if the corresponding DTF contains the WORKA=YES operand, indicating that the records are to be processed in a work area. You must define the work area in your program. You may do this by using a DS instruction, for example.

Specifying a work area causes GET to move each individual record from the input area to the work area.

If the operand is specified, all GETs for the named file must use either a register or a workname.

The workname operand is not valid for the 3881. Also, you cannot specify the WORKA operand in the DTFCD for the 3881.

GETIME (Get the Time) Macro



Requirements for the caller:

AMODE: 24 or 31

RMODE: 24 or ANY

ASC Mode: Primary

The macro obtains the time-of-day at any time during program execution. STANDARD and LOCAL are assumed if no operands are given.

If the macro is issued without an operand, the macro must not contain a comment unless the comment begins with a comma.

As long as no DATE job control statement is supplied or CLOCK=YES is specified, the calendar date and the system date in the communication region are updated every time GETIME is issued. However, when the job stream contains a DATE job control statement, only the system date in the communication region is updated when GETIME is used; the calendar date is not changed in that case.

STANDARD | BINARY | TU | MIC

If you specify:

STANDARD

The system returns the time-of-day in register 1 as a packed decimal number of the form **hhmmss**, where:

hh = Hours
mm = Minutes
ss = Seconds

with the sign in the low-order half-byte. The time-of-day may be stored and unpacked or edited.

BINARY

The system returns the time-of-day in register 1 as a binary number of seconds.

TU The system returns the time-of-day in register 1 as a binary number of units of 1/300 seconds.

MIC

The system returns the time-of-day in registers 0 and 1 as a binary number of microseconds. Bit 51 of the register pair indicates one microsecond. The specification of MIC forces the GMT operand.

LOCAL | GMT

Specify LOCAL to obtain the local time or GMT if, in your program, you want to use Greenwich Mean Time.

CLOCK=YES

Indicates that registers 0 and 1 contain, as input to GETIME, a value that was obtained by means of a STCK (store clock) instruction from the hardware time-of-day (TOD) clock:

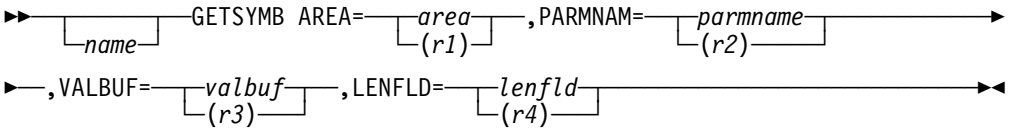
```
STCK   TOD    (Store time-of-day into 8-byte TOD)
LM     R0,R1,TOD
GETIME ...,CLOCK=YES
```

The stored value is transformed into time and date as defined by the other operands and any associated job control statements. Additionally, the date is returned in registers 14 and 15 (in the form mmddyycc or ddmmyycc, where cc indicates the century).

MFG=area | (S,area) | (r)

The MFG operand is required if the program is to be reenterable and if option STANDARD applies (with the options BINARY, TU, or MIC, reentrancy is preserved in any case). MFG specifies the address of a 64-byte dynamic storage area, that is, storage which your program obtained through a GETVIS macro. This area is required for system use during execution of the macro.

GETSYMB (Get Symbolic Parameter) Macro



Requirements for the caller:

AMODE: 24 or 31

RMODE: 24 or ANY

ASC Mode: Primary

This macro allows user programs to resolve symbolic parameters. This may be especially useful for JCL exit routines. To resolve symbolic parameters, you first have to scan the JCL commands/statements and isolate the symbolic parameters. You then invoke the macro GETSYMB to get the value of a symbolic parameter.

AREA=area | (r1)
Specifies the address of a 100-byte work area which is used as control block for saving all information related to the macro call.

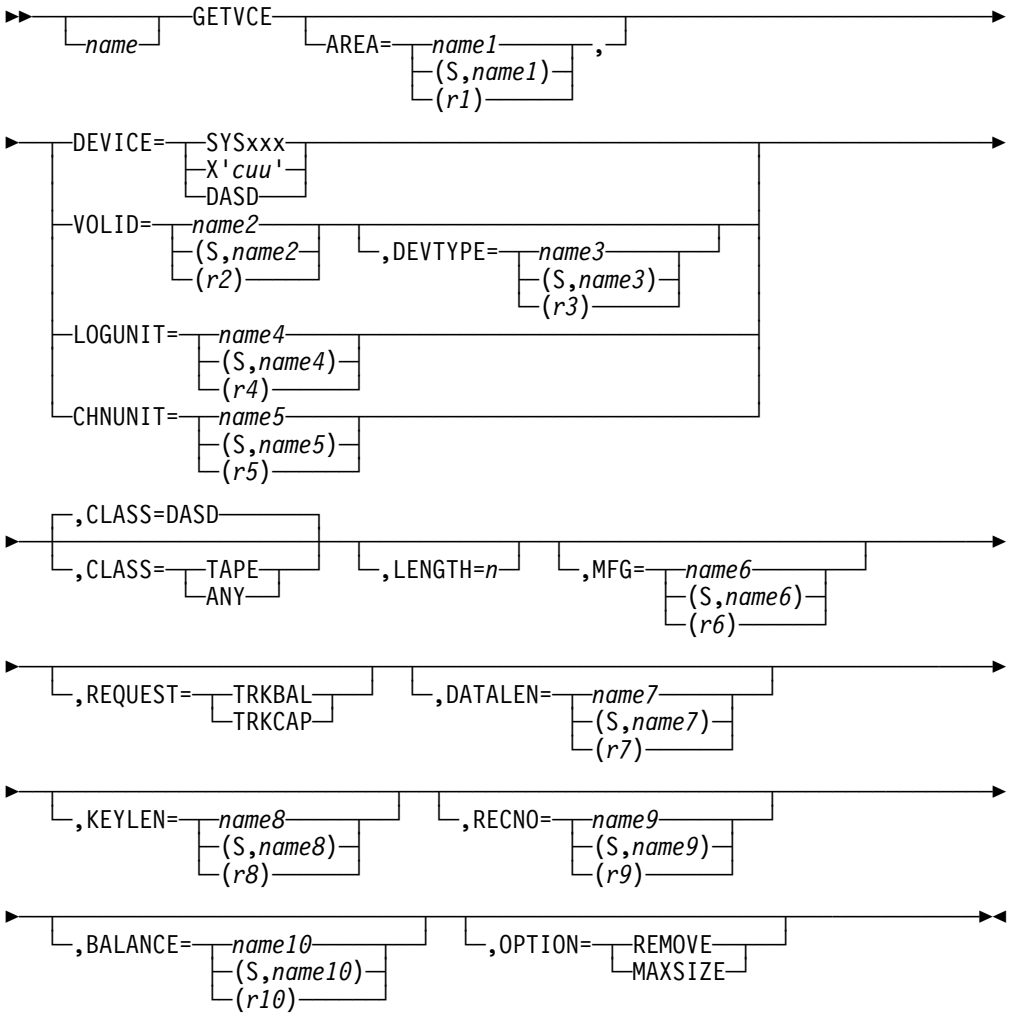
PARMNAM=parmname | (r2)
Specifies the address of a 7-byte field containing the symbolic parameter name. A name shorter than 7 bytes must start in the leftmost position and the unused bytes must be blank.

VALBUF=valbuf | (r3)
Provides the address of a buffer which will receive the character string that was defined in a previous // SETPARM job control statement for the symbolic parameter. Since this value can be up to 50 characters long, the length of the buffer must be 50 bytes.

LENFLD=lenfld | (r4)
Specifies the address of a 2-byte field where the system will provide the length of the value.

Return code 0 in register 15 indicates that the request was successful and that the symbol was found. Return code X'10' indicates that the symbol was not found. Registers 13, 14, 15, 0, 1 are destroyed.

GETVCE (Get Volume Characteristics) Macro



Requirements for the caller:

AMODE: 24 or 31

RMODE: 24

ASC Mode: Primary

The GETVCE macro returns specific volume characteristics and information about the track capacity or track balance of the specified device. Refer also to “AVRLIST and DCTENTRY” on page 215.

AREA=name1 | (S,name1) | (r1)
Specifies the address of an area where you want the volume characteristics to be stored. For the format of this area, see “GETVCE Output” on page 214.

DEVICE=SYSxxx | X'cuu' | DASD
Identifies the device whose volume characteristics are to be returned. The specification may refer to a logical unit (SYSxxx) or to a physical unit (cuu). DASD indicates that the cuu and VOLID for all DASDs is to be returned.

VOLID=name2 | (S,name2 | (r2)
Specifies the address of a six-byte field that contains the volume serial number of the device whose volume characteristics are to be returned. When register notation is used, the register must also point to a 6-byte field.

If duplicate volume serial numbers are present in the system, the first one found by GETVCE is saved and returned if the associated device is not DOWN or if all the associated devices are DOWN. In all other cases the first device which is not DOWN will be returned.

The VOLID can be qualified by the DEVTYPE operand which further limits the search.

DEVTYPE=name3 | (S,name3) | (r3)
Specifies the address of a 1-byte field that contains the device type code of the device to be returned. Specified in conjunction with VOLID, DEVTYPE will return volume and device information for the volume with the matching volume serial number and device type (for example, 3380).

LOGUNIT=name4 | (S,name4) | (r4)
Specifies the address of a 2-byte field that contains the logical unit name of the device to be returned.

CHNUNIT=name5 | (S,name5) | (r5)
Specifies the address of a 2-byte field that contains the physical unit address (channel, unit) of the device to be returned.

CLASS=TAPE | DASD | ANY
Indicates the device class for which the request is to be made. Default is DASD (disk or diskette), ANY includes TAPE and DASD.

LENGTH=n
Specifies the length of the data to be placed into the AREA field. The AREA field is initially cleared for this length before the requested information is moved in. Default is the maximum defined length.

MFG=name6 | (S,name6) | (r6)
Specifies the address of a dynamic storage area that is to be used for parameter list construction for re-entrant programs.

REQUEST=TRKBAL | TRKCAP
TRKBAL requests the *track balance* to be returned, that is, the number of remaining bytes on a track of the specified DASD. The output is returned in register 0.

Balance calculation determines the amount of physical data fitting on a track if a record were to be written or deleted. (Of course, no actual record is being written or deleted; the calculation merely gives the number of bytes left on a track if such an operation had taken place.) The operation itself is indicated in the OPTION operand: REMOVE indicates that the record is to be deleted. If OPTION has been omitted, the record is to be written.

You may provide the track balance yourself (in the BALANCE operand), which is the balance that is true **before** the operation indicated by REMOVE takes place. The value returned by GETVCE will then be the balance after the operation took place.

In the following example, the record is assumed to be removed (OPTION=REMOVE), the record to be removed is record no. 3 (RECNO=), and the balance is provided by the user (BALANCE=):

```
|<-------------Total usable space on track----------->|
<######>         =  Record size
<%%%%...%%%>     =  User-provided balance
<====...=====>   =  New balance, returned by GETVCE

Situation on entry to GETVCE:
|<######><######><######><%%%% Given by user %%%%%%%%>|

Situation on exit:
|<######><######><====== Returned by GETVCE ========>|
```

In the next example, the record is assumed to be written (no OPTION=), the record to be written is record no. 3 (RECNO=), and no balance is provided by the user:

```
Situation on entry to GETVCE:
|<######><######><.................................>|

Situation on exit:
|<######><######><######><=========================>|
```

TRKCAP requests the *track capacity* to be returned, that is, the number of whole records that will fit either into a user-supplied track balance (BALANCE=) or into the system-calculated track balance (remainder of the track). The output is returned in register 0.

The calculated value depends on the REMOVE option: If the option is not set, GETVCE takes the user-provided balance **without** subtracting the record's size from this value. (This is different from the above balance calculation, where the record size was subtracted.) This approach is useful if you want to know how many records would fit into a given balance.

DATALEN=name7 | (S,name7) | (r7)
Specifies the address of a two-byte field containing the length of one fixed-length data record. This field is processed as an unsigned binary value.

KEYLEN=name8 | (S,name8) | (r8)
Specifies the address of a one-byte field containing the key length of one fixed-length data record. This field is processed as an unsigned binary value. If non-keyed records are processed, this byte must either be set to zero or the operand must be omitted.

RECNO=name9 | (S,name9) | (r9)
Specifies the address of a one-byte field containing the record number. A record number of zero (which is also the default) results in the maximum track balance being returned to the user or being used for track capacity calculations.

BALANCE=name10 | (S,name10) | (r10)
Specifies the address of a two-byte field containing the track balance that is to be used for calculation. This balance field is processed as an unsigned binary value. If the balance is not known, this operand must be omitted.

OPTION=REMOVE | MAXSIZE

REMOVE indicates that the given record with the specified DATALEN and KEYLEN is assumed to be removed from the track by the user. GETVCE processing will calculate and/or increment the track balance or track capacity. If the user also provided a balance, it must always equal the number of bytes available on the track before the record was removed.

If the operand is omitted, the given record is assumed to be **written** on the track of the specified device.

MAXSIZE specifies that the length of the largest data record that would still fit onto the remainder of the track is to be returned (keylength has already been taken into account).

If the record to be written does not fit onto the track, return code X'24' is set, where the returned value is the amount of user data that would still fit onto the remainder of the track if the record is not written.

GETVCE Output

The output field pointed to by AREA is initially cleared for the specified length (LENGTH=) before the requested information is moved in. The output is described by a DSECT generated by the macro AVRLIST DSECT=YES,DEVICE=YES (see “AVRLIST and DCTENTRY” on page 215).

For REQUEST=TRKCAP, register 0 contains the number of whole records that will fit on the remainder of the track (input track balance).

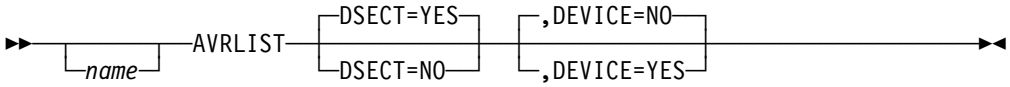
For REQUEST=TRKBAL, register 0 contains the updated track balance if a new record fits or an old record is removed. If a whole record would not fit (R15=36) and MAXSIZE was requested, register 0 is set to the maximum number of data bytes that would fit onto the remainder of the track (key bytes have already been taken care off). If a whole record would not fit and MAXSIZE was not specified, register 0 is set to zero.

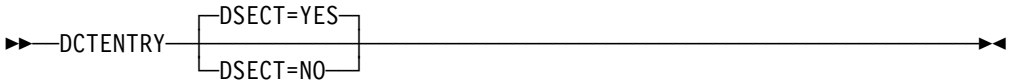
Return Codes in Register 15

0 Successful completion.
4 Successful completion, but some data is not valid.
8 The volume specified is not mounted, or the logical unit specified is not assigned, or the specified unit is not included in the system.
12 The logical unit specified is assigned IGNORE.
16 The device is not operational.
20 The parameter list is invalid (for example, the logical unit number is too high).
24 The given logical unit or device does not belong to the class specified in the CLASS operand.
28 The device is not ready.
36 For REQUEST=TRKBAL or TRKCAP only: The input balance is not sufficient to accommodate a record of the specified key and data length. MAXSIZE was specified and at least one byte of data could be written. Register 0 returns the maximum number of data bytes that would fit onto the remainder of the track.

AVRLIST and DCTENTRY

The AVRLIST macro describes volume characteristics, the DCTENTRY macro (called within AVRLIST) describes device characteristics retrieved with the GETVCE macro.





DSECT=YES | NO

YES indicates that a mapping DSECT is to be generated. NO indicates that inline code is to be generated.

DEVICE=NO | YES

YES indicates that the macro DCTENTRY is to be called within AVRLIST, thus showing the complete (volume and device) output within one DSECT.

The layout of the DSECT is described in Figure 18.

Figure 18 (Page 1 of 2). GETVCE Output Information

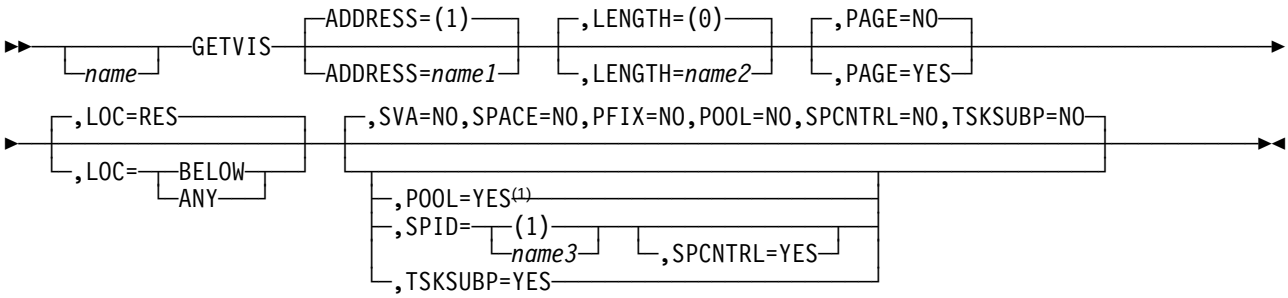
Field Name	Length	Field Description
AVRADR		Start of AVRLIST: Volume characteristics
AVRPUB	4	Address of PUB
AVRVOLID	6	Volume identifier
AVRFLAG	1	Flag byte
AVRTYPE	1	Format of device characteristics
AVRVTOC	5	DASD address of VTOC
AVRVCC	2	CKD cylinder number
AVRVHH	2	CKD track number
AVRVR	1	CKD record number
AVRCDCST	1	CKD device status
AVRVCI	1	VTOC: blocks per CI
AVRVNUM	4	VTOC: starting block number
AVRFDCST	1	Dual copy status
AVRLNO	2	Logical unit, actual value
AVRDEVC		Device type characteristics
		DCTENTRY
DCTADR		Start of DCTENTRY: Device characteristics
DCTPUBC	1	PUB device type code
DCTDTFC	1	DTF device type code (as in SECTVAL macro)
DCTUCBC	4	Unit code (as in VSAM catalog record)

Figure 18 (Page 2 of 2). GETVCE Output Information

Field Name	Length	Field Description
DCTUFLG	1	Unit I/O flags
DCTUOPT	1	Unit features
DCTUDCL	1	Unit device class
DCTUTYP	1	Unit type
DCTPCYL	2	Number of cylinders/blocks
DCTACYL	2	Cylinders/blocks in alternate area
DCTTCYL	2	Tracks per cylinder
DCTBTRK	4	Blocks per track
DCTTFIX	4	Number of blocks
DCTMAXR	2	Block size
DCTROH	1	Data+key overhead for all records
DCTROH1	1	Data+key overhead for non-last records
DCTROH2	1	Data+key overhead for last record
DCTKYOH	1	Key overhead
DCTTFLG	1	Capacity/balance calculation ID
DCTTFAC	2	Tolerance factor
DCTBYSEG	1	Bytes/segment
DCTDCBYT	1	Data correction bytes
DCTRPSC	1	RPS device type code
	1	Reserved
DCTEXTCD	6	External device type code

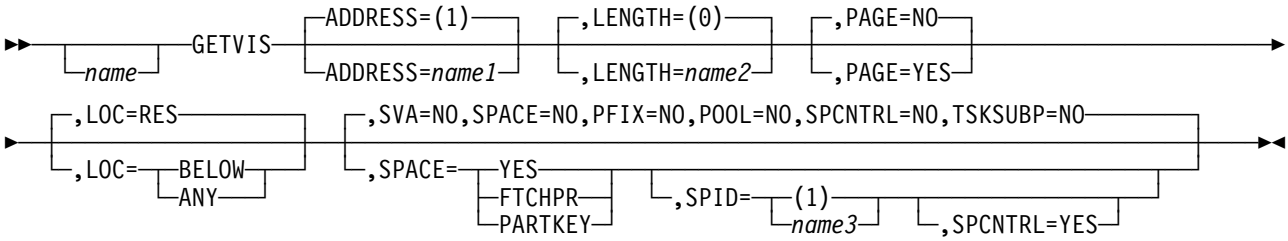
GETVIS (Get Virtual Storage) Macro

Format 1: Obtaining Storage from the Partition GETVIS Area

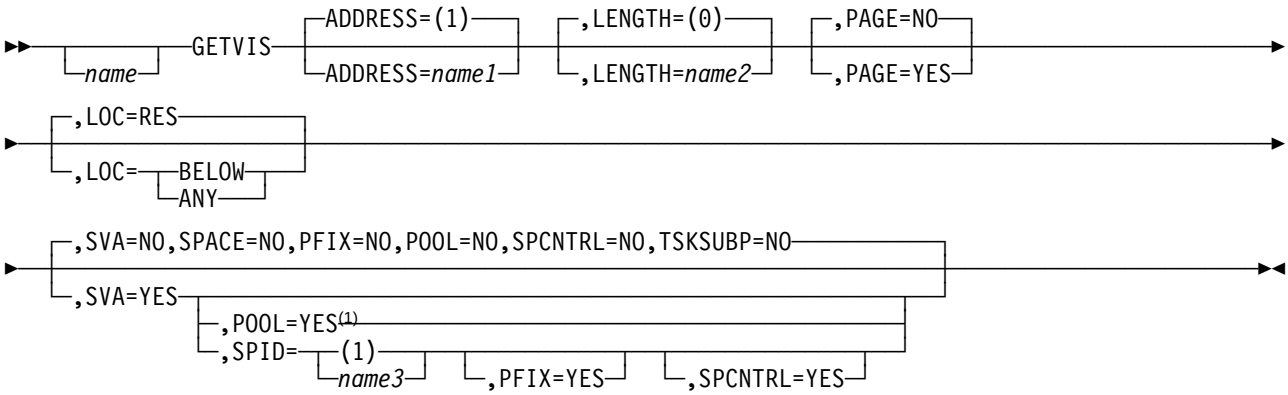


Note:
[1] POOL=YES is invalid if LOC=ANY is specified.

Format 2: Obtaining Storage from the Space GETVIS Area



Format 3: Obtaining Storage from the System GETVIS Area



Note:
[1] POOL=YES is invalid if LOC=ANY is specified.

Requirements for the caller:

AMODE: 24 or 31

RMODE: 24 or ANY

ASC Mode: Primary

The macro retrieves a block of virtual storage either from the GETVIS area of your partition or of the SVA or from the dynamic space GETVIS area.

Initially, the GETVIS area is cleared to binary zeros, and after each FREEVIS request the returned storage is cleared again. However, a program may erroneously store data into freed storage that is still addressable; this may result in a later GETVIS request for uncleared storage.

If you code the macro without any operand, the system assumes that the length of the desired virtual storage area is contained in register 0 and returns the start address of the area it retrieved for you in register 1. If the macro is issued without an operand, the macro must not contain a comment unless the comment begins with a comma.

ADDRESS=name1 | (1)
The start address of the requested virtual storage area is returned by the system either in the 4-byte field specified as a symbol by name1 or in the specified register. Register 15 must not be used because it contains the return code. The returned address is valid only if the return code in register 15 is zero. If the operand is omitted, the address is returned in register 1 only.

LENGTH=name2 | (0)
The length of the requested storage block may be specified either in the 4-byte field (specified as a symbol by name2) or in the specified register. The length is specified in number of bytes. The smallest unit that can be requested by GETVIS is either of the following:

128 bytes if the GETVIS area is part of a partition or
16 bytes if the GETVIS area is part of the SVA or of the dynamic space GETVIS area.

If the specified length is not a multiple of 128 or 16, respectively, it is rounded to the next higher multiple of 128 or 16. If the operand is omitted, the system assumes that you have specified the length in register 0.

LOC=RES | BELOW | ANY
Indicates the location of the virtual storage obtained by the GETVIS request.

RES indicates that the location of the requested virtual storage depends on the location of the caller. If the caller resides below 16M, virtual storage is to be allocated below 16MB (LOC=RES is treated as LOC=BELOW). If the caller resides above 16MB, virtual storage can be allocated anywhere (LOC=RES is then treated as LOC=ANY).

Note: The location of the caller is determined by the address portion of the PSW. Since a phase may cross the 16MB line, LOC=RES requests may return GETVIS areas above and below 16MB within the same phase.

BELOW indicates that virtual storage is to be allocated below 16MB. The allocation of storage is done from bottom to top. For *partition* GETVIS

requests, storage is allocated up to the 16MB line. For *system* GETVIS requests (SVA=YES), the area is reserved in the 24-bit system GETVIS area only, even if the 31-bit system GETVIS area is located partly or totally below 16MB.

ANY indicates that virtual storage can be allocated anywhere. For *partition* GETVIS requests, the system tries to allocate virtual storage above 16MB. If the attempt fails (or if no area above 16MB is available), the system tries to allocate virtual storage below 16MB. If this attempt also fails, no storage is allocated. Storage is always allocated from top to bottom and may cross the 16MB line. LOC=ANY is treated as LOC=BELOW if the total GETVIS area is located below 16MB.

For *system* GETVIS requests (SVA=YES), the system tries to allocate virtual storage in the 31-bit system GETVIS area, even if this area is located partially or totally below 16MB. If the attempt fails, the system tries to allocate storage in the 24-bit system GETVIS area. If there is no 31-bit system GETVIS area, LOC=ANY is treated as LOC=BELOW.

For *dynamic space* GETVIS requests (SPACE=YES), a LOC=ANY request is treated as LOC=BELOW, since there is only a 24-bit dynamic space GETVIS area. The same is true if LOC=RES is specified and the caller resides above 16MB.

Subpool Processing: Both LOC=ANY and LOC=BELOW can be given for the same subpool, so a subpool can contain areas below and above 16MB. Normally, the system tries to find the requested area within an existing subpool. However, for LOC=ANY requests, the system tries to allocate the requested storage totally above 16MB (partition) or in the 31-bit system GETVIS area. It is only when there is not enough storage available above 16MB (partition) or in the 31-bit system GETVIS area that the area below 16MB (partition) or the 24-bit system GETVIS area is searched for. (If the SPID operand is omitted, virtual storage is retrieved from a general GETVIS subpool.)

PAGE=NO | YES

PAGE=YES causes the requested storage area to start on a 2K boundary if the requested size is not larger than 2K, or on a 4K boundary if the requested size is larger than 2K. This may reduce the number of page faults, but increases storage fragmentation.

PFIX=NO | YES

If PFIX=YES is specified, the requested storage area will be PFIXed. PFIX=YES is allowed only if the SPID operand is specified and SVA=YES.

If PFIX=YES is specified, VSE backs virtual pages that reside below 16MB with real storage below 16MB, and virtual pages that reside above 16MB with real storage that can be located anywhere (preferably above 16MB).

PFIX=YES is not allowed with SPACE=YES.

POOL=NO | YES

If POOL=YES is specified, GETVIS expects to find, in register 1, an address within the defined GETVIS area where the search is to be started.

POOL=YES must not be specified together with SPACE, SPID, or with TSKSUBP=YES.

The POOL operand is invalid when specified with LOC=ANY. It is ignored for LOC=RES requests if the caller resides above 16MB. If POOL is specified, the address in register 1 (where the search for the required area is started) is regarded either as a 24-bit or 31-bit address, depending on the AMODE of the caller. The address is ignored if it does not point to the LOC=BELOW area.

SPACE=NO | YES | FTCHPR | PARTKEY

SPACE=YES indicates that the requested storage area is to be taken from the **dynamic space** GETVIS area. The requestor needs storage protection key zero.

If the GETVIS request is for a static partition, the storage area is taken from the system GETVIS area. If the request is for a static partition and the SPID operand is omitted, the storage is retrieved from a partition-related general GETVIS subpool.

SPACE=NO indicates that the storage area is not to be taken from the dynamic space GETVIS area.

SPACE=FTCHPR has the same function as SPACE=YES, but the area will be fetch-protected, that is, require key zero for read and write operations. The operand is only allowed if the SPID operand is specified.

SPACE=PARTKEY has the same function as SPACE=YES, but the area will be protected with the key of the partition. The operand is only allowed if the SPID operand is specified.

The SPACE operand must not be specified together with PFIX=YES, POOL=YES, SVA=YES, and TSKSUBP=YES.

SPCNTRL=NO | YES

SPCNTRL=YES indicates that access to the subpool specified in the SPID operand is only allowed when the correct subpool index (which is part of the subpool ID) is specified.

When you initialize a subpool (by setting the subpool index to zero), the system creates a **controlled** subpool, that is, any further request for the subpool is only allowed if the correct subpool index is passed (independent of the SPCNTRL specification).

When you refer to an existing subpool, the index must point to a subpool with the same name as the supplied one. Otherwise a return code is passed indicating an invalid subpool index.

SPCNTRL=YES is only allowed if the SPID operand is specified.

SPID=name3 | (1)

This operand can be used to create a subpool of pages of virtual storage from the partition or dynamic space GETVIS area or from the SVA.

Each subpool is defined by an eight-byte area (the subpool ID) or a register that points to the area. This eight-byte area consists of a six-byte field containing the subpool name (which you must supply) and a two-byte **subpool index** that is set by the system and should be initialized to zero when you first create the subpool. The entire eight-byte area must be used for subsequent references to the subpool. The subpool ID field is passed to the supervisor in register 1.

The address of the subpool ID is regarded either as a 24-bit or 31-bit address, depending on the AMODE (24 or 31) of the caller.

The same subpool name may be used for a SPACE=YES request within different partitions. If the request is routed to the SVA, the system does not return a unique subpool name (the subpool name is made unique internally).

The subpool name must not begin with the character I to avoid conflicts with internal IBM subpools.

When you omit the SPID (or TSKSUBP) operand, virtual storage is retrieved from a general GETVIS subpool.

SPID must not be specified together with POOL or TSKSUBP=YES.

SVA=NO | YES

SVA=YES can be specified only in a program that runs with storage protection key zero. If SVA=YES is specified, the system retrieves the desired block of virtual storage from the system GETVIS area. If SVA=NO is specified or the operand is omitted, the system retrieves the block from the partition in which your program runs.

SVA=YES must not be specified together with SPACE or TSKSUBP=YES.

TSKSUBP=NO | YES

This operand, if specified for a subtask, indicates whether or not the requested storage area is to be taken from an exclusive task subpool, which is freed when the task is terminated. If specified for a main task, this operand is ignored.

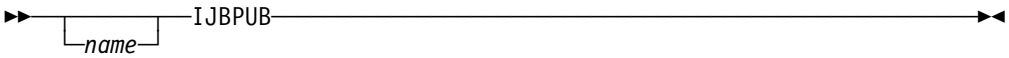
If TSKSUBP (or SPID) is not specified, GETVIS space is retrieved from the general GETVIS subpool.

TSKSUBP=YES must not be specified together with SPACE, SPID, POOL=YES, or SVA=YES.

Return Codes in Register 15

0 GETVIS completed successfully.
4 The size of the (real) partition GETVIS area is 0K.
8 The specified length is negative or exceeds the GETVIS area.
12 No more virtual storage is available in the GETVIS area, or a GETVIS request with length zero has been specified for a non-existing subpool or a subpool that has no free space.
16 The maximum number of subpools is exhausted.
20 Invalid GETVIS option.
24 Invalid subpool ID.
32 PFIX for an SVA subpool request failed.
36 An invalid subpool index was specified and (a) the request was done with SPCNTRL=YES and/or (b) the specified subpool name denotes an existing subpool that was created with SPCNTRL=YES. (A subpool index is invalid if it points to a subpool other than the supplied one. This includes a subpool index of zero for an already existing subpool.)
40 No access to the specified subpool is allowed as long as a PFIX request is pending.

IJBPUB (IJBPUB DSECT) Macro

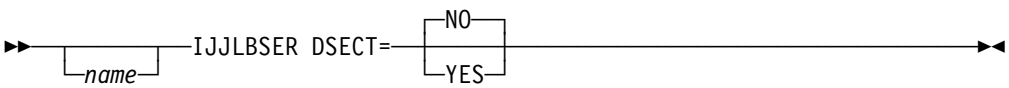


The macro generates a mapping DSECT which you need in your program to interpret the information retrieved with the macro EXTRACT ID=PUB. The generated DSECT makes fields accessible that contain:

The channel and device number
Device type code
Some device characteristics.

IJJLBSER (LBSERV DSECT) Macro

This macro maps the output area pointed to by the LBSERV SERVL field. It must always be used together with the LBSERV macro.



DSECT=NO | YES
DSECT=YES specifies that a mapping DSECT is generated. If the operand is omitted or if NO is specified, inline code is generated.

For the layout and a description of the DSECT fields, see Figure 19 on page 223.

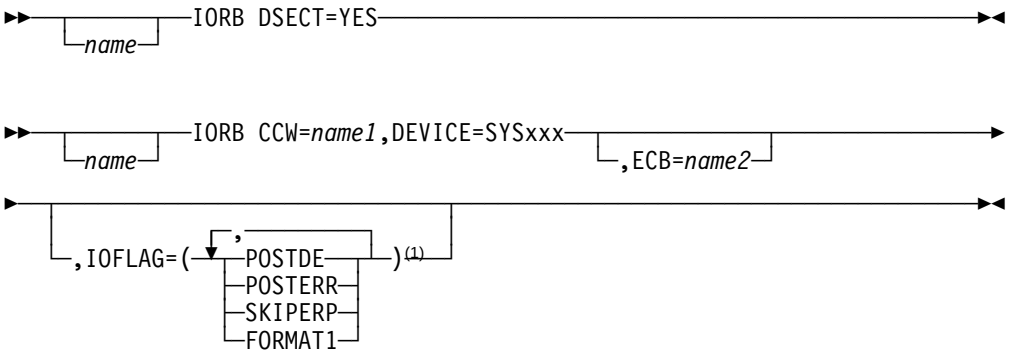
Field Name	No. of Bytes	Contents
IJJLTLEN	2	Length of area set by user
IJJLTIDN	2	Level identification
IJJLTRES	4	Reserved
IJJLTNAM	8	Name of DSECT
IJJLTRET	4	Return code
IJJLTREA	4	Reason code
IJJLFLEN	2	Length of parameter list
IJJLFREQ	2	Request type
IJJLFRWF	4	Address of Read/Write flag
IJJLFCUU	4	Address of cuu
IJJLFVOL	4	Address of volume serial number
IJJLFECB	4	Address of ECB
IJJLFLBN	4	Address of library-ID
IJJLFSCT	4	Address of source category
IJJLFTCT	4	Address of target category
IJJLFMEM	4	Address of LIBR member name
	3	Reserved
IJJLTCNT	5	Count field for number of volumes
IJJLTSTA	4	Operational status (device)
IJJLFUSW	4	User word
IJJLFPIP	4	Pointer to internal part
IJJLTMED	4	Media name of volume
	4	Reserved
IJJLTRST	44	Set to zeros
IJJLTLN		Length of DSECT

Figure 19. Layout of the LBSERV-Generated DSECT

Field IJJLTLEN must be set by the user.

IORB (I/O Request Block Definition) Macro

Use either of the formats shown below:



Note:
[1] Each option can be specified once.

Required RMODE: **24**

Note: The operands FIXLIST and FIXFLAG have become obsolete; for compatibility reasons they are, however, still accepted.

The macro generates an IORB (input/output request block). The block is generated when your program is being assembled. For the layout and contents of an IORB, see Figure 20.

Figure 20. Layout and Contents of the I/O Request Block (IORB)

Offset (In Hex)	Length (In Hex)	Contents
0	2	Residual count (same as for a CCB; see Figure 2 on page 57).
2	2	Transmission information (same as for a CCB; see Figure 2 on page 57).
4	2	CSW status bits (same as for a CCB; see Figure 2 on page 57).
6	7	Reserved.
0D	3	CSW address in CSW (same as for a CCB; see Figure 2 on page 57).
10	10	Reserved.

The block is an alternative to the CCB: instead of specifying a CCB in the EXCP macro, the address of an IORB is given.

CCW=name1
This operand gives the name of the first CCW used with the IORB. This name must be the same as the name specified in the assembler CCW statement that builds the CCW.

DEVICE=SYSxxx
This operand specifies the logical unit for the actual I/O unit with which this IORB is associated.

DSECT=YES

If the operand is specified, it should be the only one. Any other operands you specify are ignored and an MNOTE is generated by the assembler.

Specifying DSECT=YES causes the assembler to generate, as a DSECT structure, the IORB and the meaning of its fields.

ECB=name2

This operand specifies the address of the ECB to be posted when I/O is complete. The traffic bit (byte 2, bit 0) of the ECB must have been cleared before issuing the EXCP macro. The ECB area must be included in the fix list if the ECB operand is used.

Note: If FIXFLAG=(FIXED) is specified, the ECB must have be fixed in storage by a PFIX macro.

IOFLAG=(option,...)

A list of options may be specified which apply to I/O interrupt handling:

POSTDE

To indicate that device end is to be posted.

POSTERR

To indicate that an irrecoverable I/O error is to be accepted.

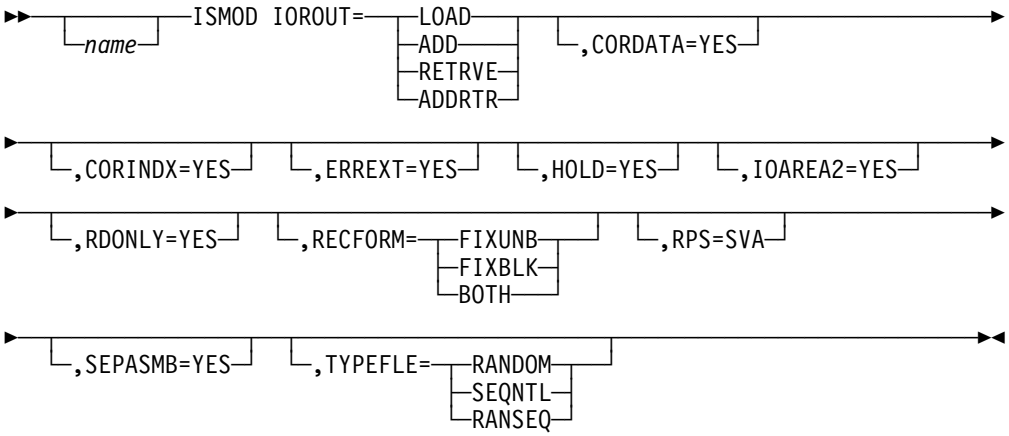
SKIPERP

To indicate that error recovery by the system is to be skipped.

FORMAT1

To indicate that the CCW is a format-1 CCW.

ISMOD (Indexed Sequential I/O Module Definition) Macro



Requirements for the caller:

AMODE: 24

RMODE: 24

ASC Mode: Primary

The macro defines a logic module for an ISAM file. If you do not provide a name for the module, IOCS generates a standard module name.

Note: If an ISMOD module precedes an assembler language USING statement or follows your program, registers 2 through 12 remain unrestricted even during assembly. However, if the module lies within your program, issue the same USING statement (as that which was issued before the ISMOD module) directly following the module. This is necessary because the ISMOD module uses registers 1 through 3 as base registers. It uses register 5 in addition if your ISMOD macro includes CORDATA=YES. Each time the module is assembled, these registers are dropped.

CORDATA=YES

Include this operand if the module is to add records to files with the IOSIZE DTFIS operand. If this operand is included, the IOSIZE operand is required in the DTF. If you omit the CORDATA=YES operand, you will not have an increase in throughput when adding records to a file.

CORINDX=YES

Include this operand to generate a module that can process DTFIS files (add or random retrieve functions) with or without the cylinder index entries resident in virtual storage. If the operand is omitted, the generated module cannot process the resident cylinder index entries.

If an irrecoverable I/O error occurs while reading indexes into virtual storage, the program will not use the resident cylinder index entries.

ERREXT=YES

Include this operand if the ERET macro is to be used with this module or if non-data-transfer error conditions are returned in filenameC. If HOLD=YES and ERREXT=YES, your program must issue the ERET macro to return to

the ISAM module to free any held tracks. See also the DTF ERREXT and HOLD operands.

HOLD=YES
This operand provides for the track hold option for both data and index records. If the HOLD operand is omitted, the track hold function is not performed.

Because track hold cannot be performed on a LOAD file, HOLD=YES cannot be specified when IOROUT=LOAD.

If HOLD=YES and ERREXT=YES, your program must issue the ERET macro to return to the ISAM module to free any held tracks.

IOAREA2=YES
Include this operand if a second I/O area is to be used, that is, if IOAREA2 is specified in any of the DTFs linked to the logic module. The operand is only valid for load or sequential retrieval functions. The module can process DTFs with one or two I/O areas specified. This operand must not be specified if TYPEFLE=RANSEQ is specified.

IOROUT=LOAD | ADD | RETRVE | ADDRTR
This operand specifies the type of module required to perform a given function:

IOROUT=LOAD
Generates a module for creating or extending a file.

IOROUT=ADD
Generates a module for adding new records to an existing file.

IOROUT=RETRVE
Generates a module to retrieve, either randomly or sequentially, records from a file.

IOROUT=ADDRTR
Generates a module that combines the features of the ADD and RETRVE modules. This module also processes any file in which only ADD or RETRVE is specified in the IOROUT operand of the DTF, and in which the TYPEFLE operand contains the corresponding specification (or a subset of it).

RDONLY=YES
This operand causes a read-only module to be generated. Whenever this operand is specified, any DTF used with the module must have the same operand.

RECFORM=FIXUNB | FIXBLK | BOTH
This operand generates a module that creates, adds to, or processes an unblocked (FIXUNB) or blocked (FIXBLK) file. If BOTH is specified, a module is generated to process both unblocked and blocked files, and the DTF may specify either FIXUNB or FIXBLK in the RECFORM operand. The RECFORM operand is required only when IOROUT specifies ADD or ADDRTR. If IOROUT specifies LOAD or RETRVE, a module that handles fixed-length blocked and unblocked files is generated, and the operand is not required.

RPS=SVA
This operand causes the RPS logic modules to be assembled.

SEPASMB=YES
Include this operand only if the module is to be assembled separately. This produces an object module ready to be cataloged into a suitable sublibrary either by the standard name or by the user-specified name. The name is used as the module's transfer address. If you omit the operand, the assembler assumes that the ISMOD macro is assembled together with the DTF in your program.

TYPEFLE=RANDOM | SEQNTL | RANSEQ
This operand is required when IOROUT specifies RETRVE or ADDRTR. RANDOM generates a module that includes only random retrieval capabilities. SEQNTL generates a module that includes only sequential retrieval capabilities. RANSEQ generates a module that includes random and sequential capabilities. It also processes any file in which the TYPEFLE operand specifies either RANDOM or SEQNTL. If TYPEFLE=RANSEQ, IOAREA2=YES must not be specified.

When all operands are omitted, the ISMOD module can only process files where IOROUT=RETRVE, TYPEFLE=RANSEQ, CORINDX, CORDATA, HOLD, and RDONLY are not specified. The name of that module is IJHZRBZZ.

Standard ISMOD Names

Each name begins with a 3-character prefix (IJH) and continues with a 5-character field corresponding to the options permitted in the generation of the module.

ISMOD name = IJHabcde

Char.	Content	Specified Option
a	A	RECFORM=FIXBLK or RECFORM=FIXUNB, IOROUT=ADD or IOROUT=ADDRTR
	B	RECFORM=FIXBLK, IOROUT=ADD or IOROUT=ADDRTR
	U	RECFORM=FIXUNB, IOROUT=ADD or IOROUT=ADDRTR
	Z	RECFORM is not specified (IOROUT=LOAD, IOROUT=RETRVE)
b	A	IOROUT=ADDRTR
	I	IOROUT=ADD
	L	IOROUT=LOAD
	R	IOROUT=RETRVE
	V	IOROUT=ADDRTR, RPS=SVA
	X	IOROUT=LOAD, RPS=SVA
c	B	TYPEFLE=RANSEQ
	G	IOAREA2=YES, TYPEFLE=SEQNTL or IOROUT=LOAD
	R	TYPEFLE=RANDOM
	S	TYPEFLE=SEQNTL
	Z	Neither is specified (IOROUT=LOAD or IOROUT=ADD)
d	B	CORINDX=YES and HOLD=YES
	C	CORINDX=YES
	O	HOLD=YES
	Z	Neither is specified
e	F	CORDATA=YES, ERREXT=YES, RDONLY=YES
	G	CORDATA=YES and ERREXT=YES
	O	CORDATA=YES and RDONLY=YES
	P	CORDATA=YES
	S	ERREXT=YES and RDONLY=YES
	T	ERREXT=YES
	Y	RDONLY=YES
	Z	Neither is specified

Subset/Superset ISMOD Names

The following chart shows the subsetting and supersetting allowed for ISMOD names. Five specifications provide for supersetting. Module IJHBABZZ, for example, is a superset of the module IJHBASZZ.

```
            +  +  +  +  +
   I  J  H  A  A  B  B  F
            B  I  R  O  O
            Z  +  +  +  +
            +  A  B  C  S
            A  R  S  Z  Y
            U  *  +     *
            Z  L  G     G
                  S     P
                  +     +
                  G     T
                  Z     Z
```

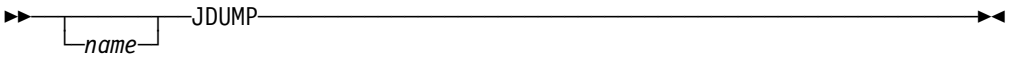
+
Subsetting/supersetting permitted.

***** No subsetting/supersetting permitted.

If two or more modules with the same entry point are included, the linkage editor message 2143I (invalid duplication of entry point label) is generated. Occasionally these entry points are not obvious when using the preceding chart, but the module can perform the indicated functions. This message can usually be suppressed by including a superset module. However, modules with and without prime data in main storage or modules with TYPEFLE=RANDOM and IOAREA2=YES cannot be combined. Therefore, you should take either of the following actions:

- Specify prime data in core for each ADD type DTF in your program. In this case, superset modules are generated.
- Specify the MODNAME operand in the DTF, and include an ISMOD of that name. The DTF then generates only the specified module.

JDUMP (Job Dump Request) Macro



Requirements for the caller:

AMODE: 24 or 31

RMODE: 24

ASC Mode: Primary

This macro provides a hexadecimal dump of the following:

- The contents of either the entire supervisor area and the used part of the system GETVIS area, or of some supervisor control blocks only. The dump includes the contents of some supervisor control blocks only if either:
  - The STDOPT job control command specifies DUMP=PART or DUMP=NO, or
  - the job is submitted with PARTDUMP or NODUMP specified in a job control OPTION statement.
- The contents of the partition that issued the macro.
- The contents of the registers.

In addition, the macro causes the job to be terminated if JDUMP was issued by the main (or only) task of the program. If JDUMP was issued by a subtask, the macro causes that subtask to be detached without terminating the program in the partition.

If the job control option SYSDUMP is active, the output of the dump is directed to the dump sublibrary of the partition. If the option NOSYSDUMP is active, the output is directed to SYSLST, which can be assigned to a printer, disk or tape unit.

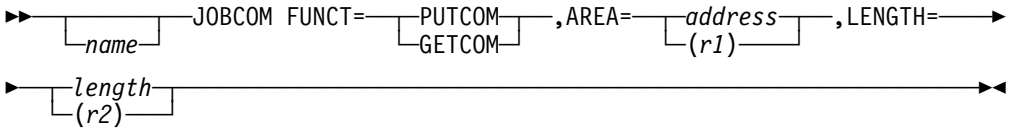
If SYSLST is assigned to tape, this tape must be positioned as desired.

If SYSLST is assigned to an IBM 3211 and indexing was used before you issue the JDUMP macro, a certain number of characters on every line of the printed dump may be lost. To avoid this, reload the printer's FCB (forms control buffer) by issuing an LFCB macro before you issue the JDUMP macro. The FCB image you load must not have an indexing byte.

If JDUMP is issued by a job running in real mode, the storage contents of the partition are dumped only up to the limit as determined by the SIZE operand of the EXEC job control statement, plus the storage obtained dynamically through the GETVIS macro. If SIZE was not specified, the entire partition will be dumped.

If JDUMP is issued by a program running in virtual mode, the entire partition is dumped.

JOBCOM (Job Communication) Macro



Requirements for the caller:

AMODE: 24
RMODE: 24
ASC Mode: Primary

The macro provides for communication between jobs or job steps in a partition. For *dynamic* partitions, this applies only to VSE jobs and job steps within one VSE/POWER job, because a dynamic partition and its VSE control blocks are only existent during the execution of a VSE/POWER job.

Information being communicated is stored in a 256-byte area. The system provides such an area for each partition. Through the JOBCOM macro, a program either moves information into that area or retrieves information that had previously been stored there by another program. The area remains unaltered from one job (or job step) to the next. Unless it is modified through execution of the JOBCOM macro, the contents of the area remain unchanged over any number of jobs. The JOBCOM macro is not reentrant.

The program that issues the JOBCOM macro must provide a register save area 18 fullwords long. Prior to execution of the macro, register 13 has to point to that save area.

Note: When the JOBCOM macro is used, the contents of registers 14 through 1 are destroyed.

FUNCT=PUTCOM | GETCOM
This operand describes the function that the macro is to perform. Specifying PUTCOM causes information to be stored into the system-supplied area. The number of bytes to be moved is given by the LENGTH operand. If LENGTH yields a value smaller than 256, the remainder of the area is left unaltered.

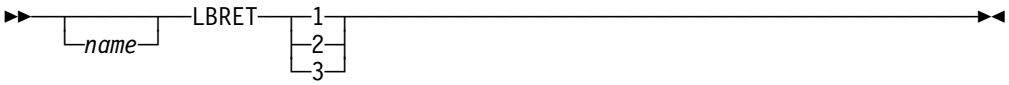
Specifying GETCOM indicates that information is to be retrieved from the system-supplied area. Again, the number of bytes to be moved is given by the LENGTH operand.

AREA=address | (r1)
This operand gives the address of a field where the program provides (FUNCT=PUTCOM specified) or receives (FUNCT=GETCOM specified) the information to be moved.

LENGTH=length | (r2)
This operand specifies the number of bytes to be moved. The value is either given as a self-defining term or in register notation. If register notation is used, the specified register is expected to contain the length value.

Length should be a positive number up to 256. If it is zero or negative, no information gets moved. If it is greater than 256, only 256 bytes are moved.

LBRET (Label-Routine Return) Macro



Requirements for the caller:

AMODE: 24

RMODE: 24

ASC Mode: Primary

The macro is issued in your subroutines when you have completed processing labels and wish to return control to IOCS. LBRET applies to subroutines that write or check user-standard on disk or on magnetic tape, write or check tape nonstandard labels, or check disk extents. The operand used – 1, 2, or 3 – depends on the function to be performed. The functions and operands are explained below.

Checking Disk Extents

When processing an input file with all volumes mounted, you can process your extent information. After each extent is processed, use LBRET 2 to receive the next extent. When extent processing is complete, use LBRET 1 to return control to IOCS.

Checking User Standard Labels on Disk

IOCS passes the labels to you one at a time until the maximum allowable number is read (and updated), or until you signify you want no more. In the label routine, use LBRET 3 if you want IOCS to update (rewrite) the label just read and pass you the next label. Use LBRET 2 if you simply want IOCS to read and pass the next label. If an end-of-file record is read when LBRET 2 or LBRET 3 is used, label checking is automatically ended. If you want to eliminate the checking of one or more remaining labels, use LBRET 1.

Writing User Standard Labels on Disk

Build the labels one at a time and use LBRET to return to IOCS, which writes the labels. Use LBRET 2 if you want control returned to you after IOCS writes the label. If, however, IOCS determines that the maximum number of labels has already been written, label processing is terminated. Use LBRET 1 if you wish to stop writing labels before the maximum number of labels is written.

Checking User Standard Tape Labels

IOCS reads and passes the labels to you one at a time until a tapemark is read, or until you indicate that you do not want any more labels. Use LBRET 2 if you want to process the next label. If IOCS reads a tapemark, label processing is automatically terminated. Use LBRET 1 if you want to bypass any remaining labels.

Writing User Standard Tape Labels

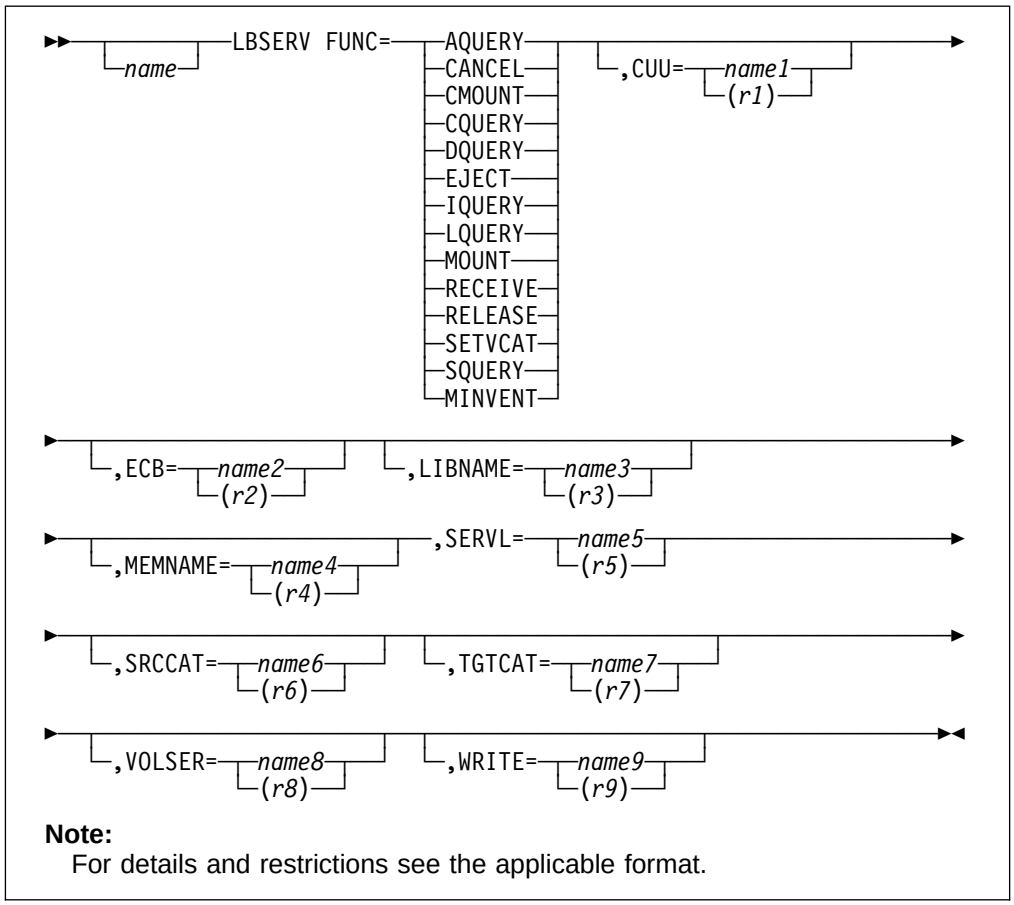
Build the labels one at a time and return to IOCS, which writes the labels. When LBRET 2 is used, IOCS returns control to you (at the address specified in the DTFxx LABADDR operand) after writing the label. Use LBRET 1 to terminate the label set.

Writing or Checking Nonstandard Tape Labels

You must process all your nonstandard labels at once. Use LBRET 2 after all label processing is completed and you want to return control to IOCS.

LBSERV (Control IBM 3494 Tape Library) Macro

Overview of LBSERV Macro



Note:
For details and restrictions see the applicable format.

LBSERV allows to pass tape handling requests to an IBM 3494 Tape Library Dataserver. These requests are routed either

- via the VSE Library Control Device Driver (LCDD) in VSE or
- via the VSE Guest Server (VGS) machine in VM

to the IBM 3494 Tape Library Dataserver.

For any LBSERV request you must use the **IJJLBSER** mapping macro. IJJLBSER must start on doubleword boundary. After each LBSERV request (except CANCEL and RECEIVE) a WAIT macro has to be issued, if the return code from the LBSERV request in register 15 is zero. After the ECB gets posted, a LBSERV RECEIVE request must be issued which provides return and reason code information in the field indicated by SERVL.

Note: Be aware that the program using the LBSERV macro must reserve GETVIS space. If the LBSERV macro is supported via the VSE Guest Server (VGS), you must provide at least 100K permanent PFIX for each partition using VGS. The following example defines the 100K permanent PFIX above the 16MB line:

```
// SETPFIX LIMIT=(,100K),PERM
// EXEC ...SIZE= must reserve GETVIS space
```

Figure 21 lists the input and output parameters required for the various function operands.

Figure 21. LBSERV : OPERANDS PER FUNCTION

	CUU	ECB	LIBNAME	MEMNAME	SERVL	SRCCAT	TGTCAT	VOLSER	WRITE
A-QUERY		i(*)	o		i	o		i	
CANCEL					i				
C-MOUNT	i	i	i b		i	i	i b	o	i
C-QUERY		i	i b		i	i b			
D-QUERY	i	i			i	o		o	
EJECT		i	i b		i			i	
I-QUERY		i	i b		i	i b			
L-QUERY		i	i b		i				
MINVENT		i	i b	i	i	i b	i		
MOUNT	i	i	i b		i		i b	i	i
RECEIVE					i				
RELEASE	i	i			i				
SETVCAT		i	i b		i	i b	i	i	
S-QUERY		i(*)	i b o		i	o		i	

```
'i' = required input parameter
'o' = required output parameter
'b' = input parameter contents may be a blank value to indicate
      optional
'*' = a free CUU is required for this request when running under
      VM (VGS)
```

Note: In order to get **CUU** for C-MOUNT, D-QUERY, and MOUNT, use the EXTRACT macro or the ASSGN command to assign the device.

The following requirements for the caller, register usage convention, input parameters and return codes apply to all LBSERV functions.

Requirements for the caller:

AMODE: 24 or 31

RMODE: ANY

ASC Mode: Primary

Register usage convention:

r(x) for register notation R(x) any general register from 2 to 12 can be used

Registers 0, 1 will be destroyed.

Register 13 must point to a 72 byte save area.

Register 14 return register.

Register 15 branch register at input, return code from LBSERV macro call on output

Input Parameters:

CUU = name1 | (r1)

address of an area containing a 4 character device number to be used for this operation.

ECB = name2 | (r2)

address of an area containing the ECB which is posted when the AQUERY request completes.

LIBNAME = name3 | (r3)

address of an 8 character area where the library name - if found - is stored.

MEMNAME = name4 | (r4)

address of an area containing a 8 character name of a VSE library member. This member contains the list of volumes to be managed by MINVENT request. Please see *IBM 3494 User"s Guide: Library Control Device Driver for VSE/ESA* for a definition of the library and sublibrary name.

SERVL = name5 | (r5)

address of an area where the return information will be stored when the request completes. This area **must** start on a double word boundary. The area is mapped by macro IJJLBSER. The area's length, IJJLTLN, must be stored into field IJJLTLEN of IJJLBSER by the user. It is checked at LBSERV invocation if its size is as large as required. The first part up to label IJJLTRST is available to the user, the rest is reserved for system use only.

SRCCAT = name6 | (r6)

address of a 10-character area where the tape volume's category (source category) will be stored. A blank name (10 blank characters) indicates to ignore the source category.

Valid category names are:

- INSERT
- MANEJECT (not for VGS)
- PRIVATE (not for VGS)
- SCRATCH
- SCRATCHnn, nn=00, ...,31 for LCDD, nn=00,...,15 for VGS (mapped to n=0,...,F)
- SCRATCHn, n=0,...,F VGS only

TGTCAT = name7 | (r7)

address of a 10-character area containing the new category name (target category) for the volume to be used for this function. The target category is assigned to the volume. Valid category names are:

- blank
- PRIVATE
- SCRATCH
- SCRATCHnn, nn=00, ...,31 for LCDD, nn=00,...,15 for VGS (mapped to n=0,...,F)
- SCRATCHn, n=0,...,F VGS only

A blank field (10 blank characters) as category name indicates to ignore the parameter and leave the category unchanged.

VOLSER = name8 | (r8)
address of an area where the six character volume serial number of the volume to be queried.

WRITE = name9 | (r9)
address of an area containing a one character read/write flag, where R means read-only and W means write. R is the default.

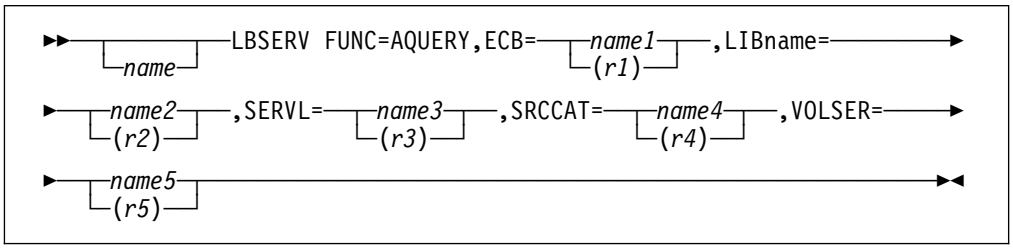
Return codes in register 15:

x'0000' request passed successfully. After ECB was posted issue LBSERV RECEIVE and check return and reason code for this request.

x'0004' request passed, but warning condition. Check reason code.

x'0005' or >x'0005'
request not successful. Check reason code.

LBSERV AQUERY



LBSERV AQUERY requests a volume's location to be verified. All libraries known to LCDD or VGS will be queried until the specified volume is found. If the volume is found, the name of its library, source category and the media type will be returned.

Output:

LIBNAME name of the library containing the tape volume

SRCCAT name of the category the tape volume belongs to

IJJLTSTA volume status, see Figure 22.

IJJLTMED media type, see Figure 23 on page 238.

IJJLTREA reason code in IJJLBSER

IJJLTRET return code in IJJLBSER

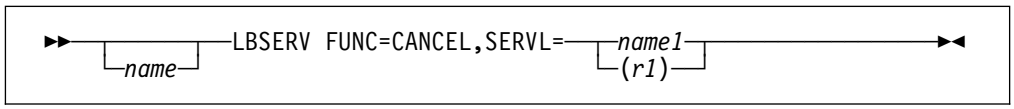
Size	Contents	Meaning
4 char.	0000	no special condition
4 char.	8000	inaccessible
4 char.	4000	mounted
4 char.	2000	queued for mount
4 char.	1000	being mounted
4 char.	0800	queued for demount
4 char.	0400	being demounted
4 char.	0200	queued for eject
4 char.	0100	being ejected
4 char.	0080	queued for audit
4 char.	0040	being audited
4 char.	0020	misplaced
4 char.	0010	missing or damaged label
4 char.	0008	used in manual mode
4 char.	0010	manually ejected

Figure 22. Volume Status in IJJLTSTA

Size	Contents	Meaning
4 char.	CST1	3490E
4 char.	CST2	3490E
4 char.	CST3	3590
4 char.	CST4	3590

Figure 23. Media Type in IJJLTMED

LBSERV CANCEL



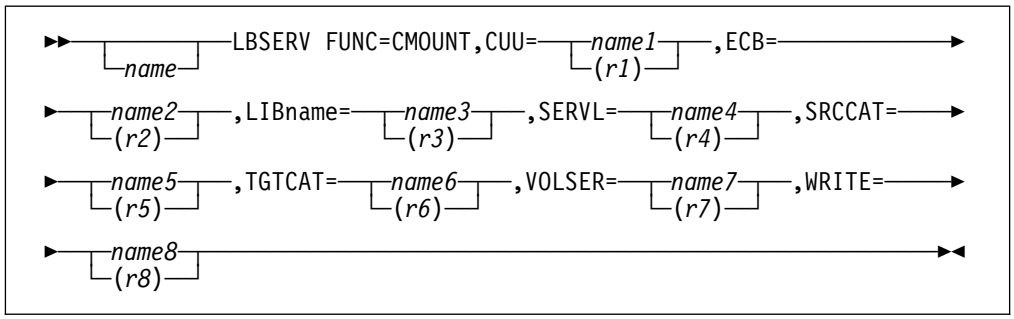
A prior CMOUNT/MOUNT request is to be cancelled because it is no longer needed. The prior request is identified by the address of the IJJLBSER area passed in SERVL which still contains CUU, VOLSER and ECB address of the last executed function.

Output:

IJJLTREA reason code in IJJLBSER

IJJLTRET return code in IJJLBSER

LBSERV CMOUNT



CMOUNT requests a volume from the specified source category to be mounted on a device in a library. If TGTCAT is specified then this target category is assigned to the mounted volume.

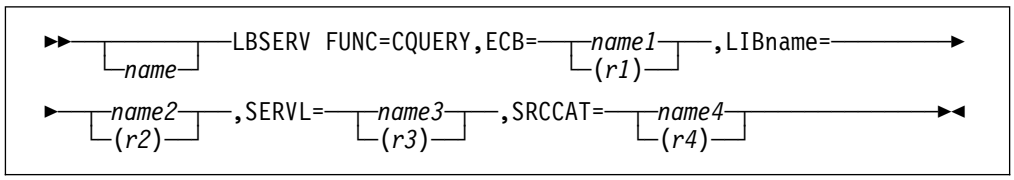
Output:

IJJLTREA reason code in IJJLBSER

IJJLTRET return code in IJJLBSER

VOLSER volume serial number of mounted tape is stored in the area pointed to by VOLSER.

LBSERV CQUERY



CQUERY requests the number of volumes in the specified category to be returned to the requestor. If SRCCAT is not specified (set to blank) then the number of all volumes within the library is returned.

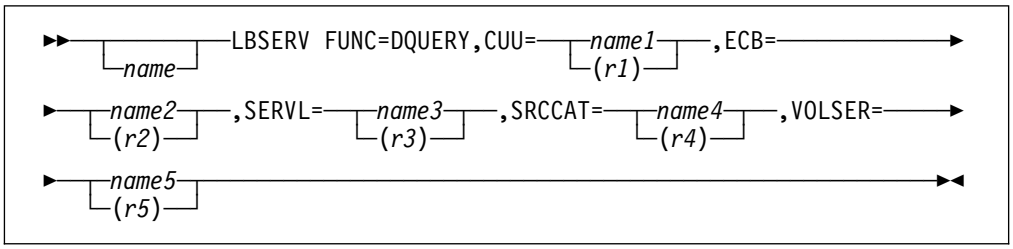
Output:

IJJLTCNT number of volumes in specified category and library

IJJLTREA reason code in IJJLBSER

IJJLTRET return code in IJJLBSER

LBSERV DQUERY



DQUERY request the status of a device in a library.

Output:

IJJLTSTA status of device as explained in Figure 24.

IJJLTREA reason code in IJJLBSER

IJJLTRET return code in IJJLBSER

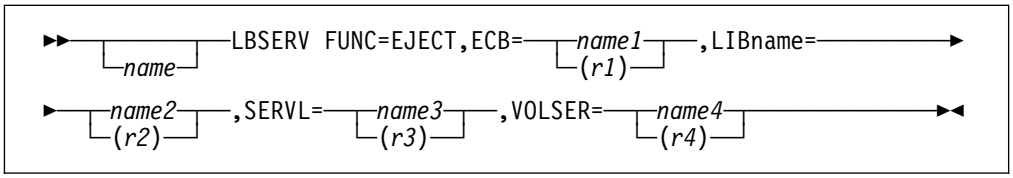
SRCCAT name of the category the tape volume belongs to if a tape is mounted. If no tape is mounted the field will be blank.

VOLSER volume serial number of the tape volume if a tape is mounted. If no tape is mounted the field will be blank.

Size	Contents	Meaning
4 char.	0000	installed and available
4 char.	8000	not installed or available

Figure 24. Device Status in IJJLTSTA

LBSERV EJECT



EJECT request to eject the specified volume from the specified library. The tape volume is moved to the Convenience I/O station if this feature is installed or to the high-capacity output area if one is defined in the 3494. If both output facilities are available the tape volume goes to the Convenience I/O.

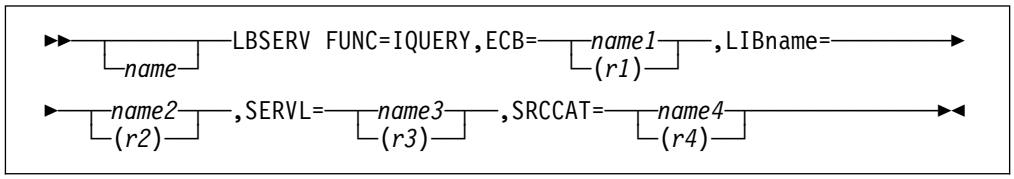
Note: SETVCAT provides an interface to specify the output area.

Output:

IJJLTREA reason code in IJJLBSER

IJJLTRET return code in IJJLBSER

LBSERV IQUERY



LBSERV IQUERY requests to build an inventory list of the volumes of the specified category of the specified library. If category is omitted (blank name) an inventory list of the entire library is built. A blank library name (8 blank characters) indicates to use the default library. The inventory list is ready for use after successful completion of the request.

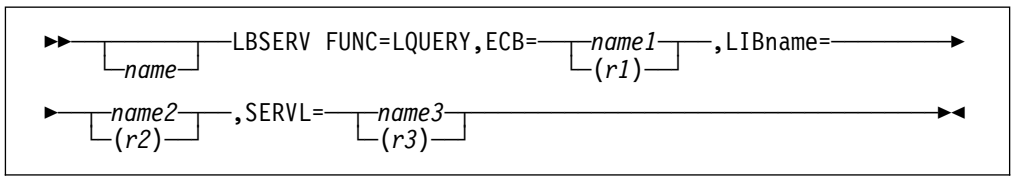
Output:

Inventory list The inventory list is stored as a member of a VSE sublibrary. See *IBM 3494 User"s Guide: Library Control Device Driver for VSE/ESA* for naming conventions for the VSE library and sublibrary, and how the member name is derived from the specified source category.

IJJLTREA reason code in IJJLBSER

IJJLTRET return code in IJJLBSER

LBSERV LQUERY



LQUERY asks for the operational status of a library.

Output:

IJJLTSTA status of the library explained in Figure 25.

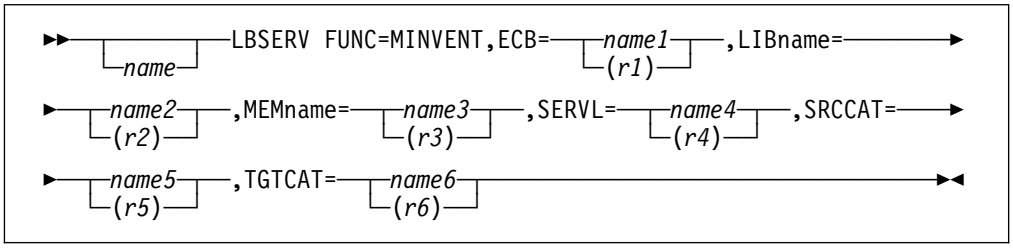
IJJLTREA reason code in IJJLBSER

IJJLTRET return code in IJJLBSER

```
Size       Contents     Meaning
4 char.    0000         automated mode
4 char.    0100         paused mode
4 char.    0200         manual mode
```

Figure 25. Library Status in IJJLTSTA

LBSERV MINVENT



MINVENT (Manage Inventory) requests that the list of volumes in the referenced member name and of the specified source category be transferred to the specified target category. Lists created by LBSERV IQUERY are pre-formatted for use as input to the LBSERV MINVENT request. Query lists are described in their own unique section of *IBM 3494 User"s Guide: Library Control Device Driver for VSE/ESA*

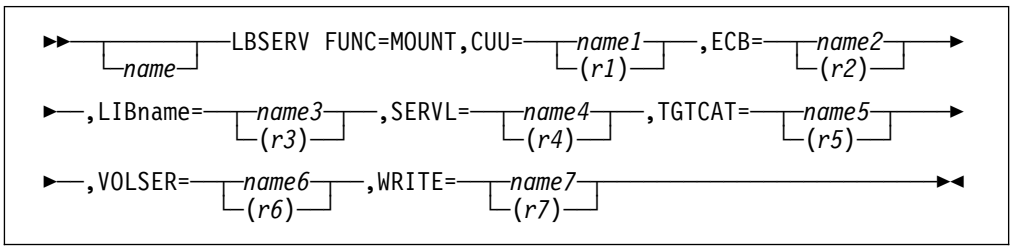
Output:

Inventory list specified in MEMNAME is updated with result of category change per volume.

IJJLTREA reason code in IJJLBSER

IJJLTRET return code in IJJLBSER

LBSERV MOUNT



MOUNT requests a volume with the specified volume serial number to be mounted on a device in a library. If TGTCAT is specified then this target category is assigned to the mounted volume.

Output:

IJJLTREA reason code in IJJLBSER

IJJLTRET return code in IJJLBSER

LBSERV RECEIVE

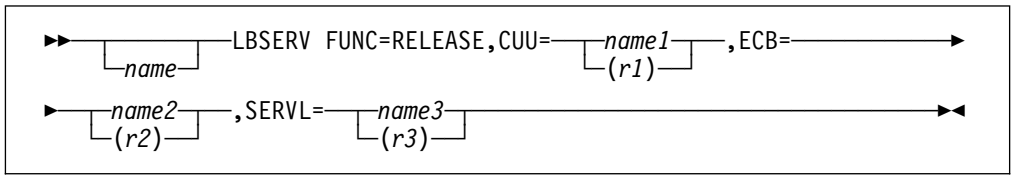
LBSERV RECEIVE must be issued after the ECB of any functional request was posted, except for FUNCtion CANCEL. It retrieves the return information from the previously issued request. All input from the last executed function is left in the IJJLBSER area pointed to by SERVL.

Output:

IJJLTREA reason code in IJJLBSER

IJJLTRET return code in IJJLBSER

LBSERV RELEASE



LBSERV RELEASE frees a cuu used in a prior MOUNT request, no longer needed by the VSE job. A rewind unload is issued for the specified device. The actual volume on the device will be returned to the library. Note that mount ownership can only be reset by EOJ processing.

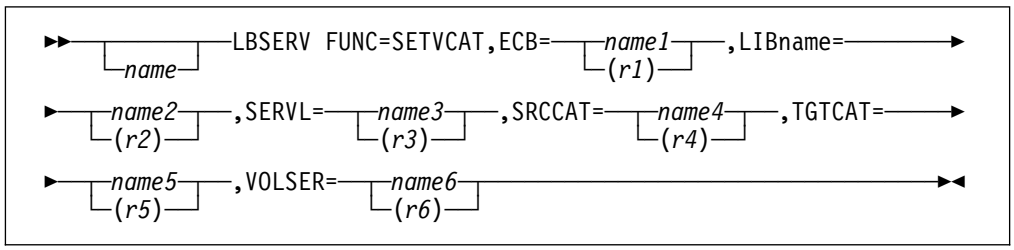
For VGS support: VM will release the real device attached at the specified virtual address so that it may be used by another VM guest, if not dedicated.

Output:

IJJLTREA reason code in IJJLBSER

IJJLTRET return code in IJJLBSER

LBSERV SETVCAT



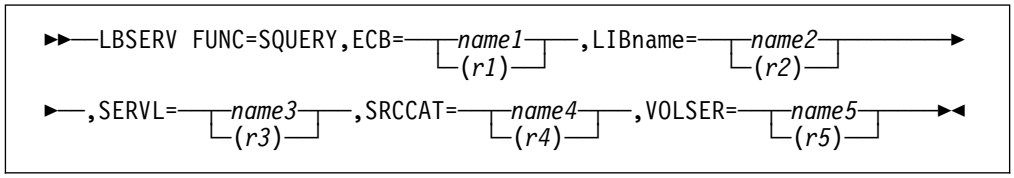
LBSERV SETVCAT assigns a category to a volume. If SRCCAT is specified, the volume has to be of this category for the assignment to be changed.

Output:

IJJLTREA reason code in IJJLBSER

IJJLTRET return code in IJJLBSER

LBSERV SQUERY



LBSERV SQUERY requests a volume's location to be verified. A single query in the specified library is performed. If LIBNAME is not specified (set to blank) the default

library is queried. If the volume is found its source category and media type are returned, and in case of the default library also the library name.

Output:

IJJLTSTA volume status, see Figure 22 on page 237.

IJJLTMED media type, see Figure 23 on page 238.

IJJLTREA reason code in IJJLBSER

IJJLTRET return code in IJJLBSER

LIBNAME only for default library (blank name on input): name of library where volume was found

SRCCAT name of the category the tape volume belongs to

Reason Codes

Reason codes are stored in field IJJLTREA. They indicate the source of the problem and of the additional return code stored in field IJJLTRET. Please note that both, reason and return code, are stored in character format.

Overview of the originating area:

If the reason code is

< **C'4000'** error detected by RMS (refer to *DFSMS/VM Removable Media Services User''s Guide and Reference*, SC35-0141), or LCDD (refer to *IBM 3494 User''s Guide: Library Control Device Driver for VSE/ESA*, GC35-0176).

> **C'5000' and** < **C'6000'** error detected by VGS (refer to Figure 26) or LCDD (refer to *IBM 3494 User''s Guide: Library Control Device Driver for VSE/ESA*).

> **C'6000'** error detected by LBSERV macro (refer to Figure 27 on page 246).

Figure 26 (Page 1 of 4). Reason Codes Generated by VGS

Reason	Meaning	Explanation/Action
5000	Request not authorized	The authorization user exit has determined that the request is not authorized.
5003	Insufficient data length	The length of the request data was smaller than the minimum required number of bytes. This is an internal program error; contact IBM service.
5008	Invalid request type	An unrecognized request type was specified. This is an internal program error; contact IBM service.
5011	Request to be cancelled not found	A request to cancel a mount request cannot be completed successfully, because the request is not longer in-process on the VGS machine. Issue rewind/unload to queue a dismount operation in the library.

Figure 26 (Page 2 of 4). Reason Codes Generated by VGS

Reason	Meaning	Explanation/Action
5017	Request cancelled by a cancel-mount request	This request is the target request for a cancel-mount operation. The reason code in the response message for the original mount request. The mount operation was not completed.
5020	Request cancelled by an immediate machine shutdown	This request is purged from the VGS in-process files during immediate shutdown processing of the VGS machine. The mount operation. may have completed successfully, but the status is not known to VGS at the time of shutdown.
5023	Device not found	The device specified for release is not found in the VGS device-use file. No action is taken.
5026	Library not known	The library name specified in a request is not found in the VGS library configuration file LIBCONFG LIST A. Update or correct this file and retry the request.
5028	Device information not obtained	VGS is not able to get information about real tape drives currently attached to the requesting VSE guest. The request cannot be processed. Ensure that the VGS is machine has privilege-class B authorization in its directory record.
5029	Problem with CP ATTACH or DETACH operation	VGS is not able DETACH a device from the requesting guest or to re-ATTACH a guest device to itself for performing the requested function. Ensure that the VGS is machine has privilege-class B authorization in its directory record.
5030	Virtual cuu not provided	A mount, release, cancel, or query device operation has not specified the required virtual cuu field.
5031	Virtual cuu contains invalid characters	A mount, release, cancel, or query device operation does not use valid numeric or hex-digit characters.
5032	Volume label not provided when required	A mount, query volume, cancel,eject, or set volume category operation has not specified a volume serial number.
5033	Source category not provided when required	A mount from category operation has not specified a source category name.
5034	Target category not provided when required	A set volume category or manage inventory operation has not specified a target category name.
5035	Member name not provided when required	A manage inventory operation has not specified a member name for the source file containing volume category assignments.
5040	Error reading result of Query or Manage Inventory request	The VGLIBSRV machine was not able to read the request final status information from the secondary inventory support machine's 191 disk. The result of the inventory operation is unknown.

Figure 26 (Page 3 of 4). Reason Codes Generated by VGS

Reason	Meaning	Explanation/Action
5041	Timeout occurred during Query or Manage Inventory request	The VGLIBSRV machine did not receive final status on the request within the time limit specified by the INV_MAX_TIME customization variable. The request may still complete successfully, or the secondary machine may have encountered an error. Check the status of the secondary inventory support machine.
5042	Secondary inventory server is not logged on	The VGS machine has determined the there the secondary inventory support server is not logged on, or its logon status is cannot be obtained. The inventory request cannot be processed.
5043	Secondary inventory server's 191 is not linked by VGLIBSRV	Inventory requests cannot be processed by VGS unless the secondary support server is defined, logged on, and it's 191 disk is linked by VGLIBSRV.
5044	Secondary inventory server is not identified to VGLIBSRV	The customization variable HELPER_ID in the customization exec does not specify a valid server name. Inventory requests cannot be processed.
5080	Locked failed, Librarian member already locked	The Librarian member specified in a Manage Inventory request or implied by category-inventory naming conventions for a Query Inventory request cannot is already locked. Member is locked by another user, wait and resubmit the request.
5081	Lock failed, fully-qualified Librarian member is not valid	The Librarian member specified in a Manage Inventory request could not be locked and most likely does not exist. Ensure that the Library and Sublibrary names specified in the customization exec are valid and that the specified member already exists in the sublibrary.
5082	Unspecified lock failure for Librarian member	The Librarian member specified in an Inventory request could not be locked for some unknown reason. This is most likely an internal error.
5083	Librarian write failure	The Librarian member specified in an Inventory request could was not successfully written in the Librarian file in the VSE guest machine.
5084	RMS error on Query Library Inventory command	An unspecified error occurred requesting the inventory report from the RMS machine. Retry the request. If the problem persists, logon to the secondary inventory support machine and check the console during request submission, and/or the RMS Master console.
5085	Invalid source category specified for Query Inventory	The source category requested for inventory in not supported or is not a valid category name.

Figure 26 (Page 4 of 4). Reason Codes Generated by VGS

Reason	Meaning	Explanation/Action
5086	LIBRCMS server name not found for the requesting VSE guest	When multiple VSE guests are supported by VGS, a LIBRCMS server cross-reference file on the secondary inventory support machine 191 disk must provide the name of the LIBRCMS server for each VSE guest. This no server was found for the requesting VSE guest.
5090	Secondary inventory support internal error - Erase	The CMS ERASE command failed for a file in the inventory support server's 191 disk.
5091	Secondary inventory support internal error - Disk I/O	The EXECIO command failed for a file in the inventory support server's 191 disk. Check for disk-full condition.
5092	Secondary inventory support internal error - Copyfile	The CMS COPYFILE command failed for a file in the inventory support server's 191 disk. Check for disk-full condition.
5093	Secondary inventory support internal error - Query Reader	The Query Reader command failed checking the reader for returned inventory report.
5094	Secondary inventory support internal error - Receive	The Receive command failed reading the report file from the reader to the 191 disk. Check for disk-full condition.
5095	Secondary inventory support internal error - Unexpected disk files	At least one previous inventory report was detected on the secondary inventory server's 191 disk. The integrity of inventory data cannot be guaranteed. Report is not updated in the VSE Librarian file.
5096	Secondary inventory support internal error - Mismatch	The name of the category inventoried on the RMS inventory report does not match the source category name in this request. The report is not updated in the VSE Librarian file.

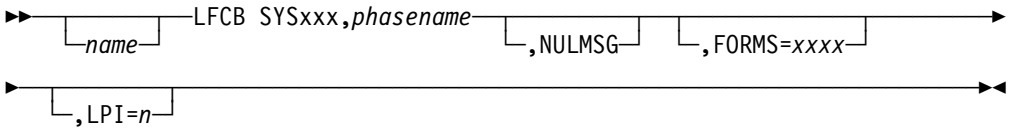
Figure 27 (Page 1 of 2). Reason Codes Generated by VSE

Reason	Meaning	Explanation/Action
61xx	XPCC internal error	An XPCC internal error occurred, please check the return code and reason code coming from XPCC, where xx in reason code 61xx is the reason code from XPCC. Possible return codes are: 0038 (return code) 1. The maximum of possible XPCC connections may have been reached or the APPC connection is not available. Please try later. 2. Check the SYS command in your IPL procedure. SYS=VM must specified if VGS is used, SYS=VSE must be specified if LCDD is used. 0039 (return code) Please check that you provided at least 100K permanent PFIX for this partition using the SETPFIX command. If the error persists, please contact IBM service.
62xx	APPC internal error	An APPC internal error occurred, maybe the XPCC/APPC connection went down. Please check the return code and reason code coming from APPC, where xx in reason code 62xx is the reason code coming from APPC. If the error persists, please contact IBM service.
63xx	LBSERV failure	An error in LBSERV code occurred. Please check the reason code xx where xx means: 01 The version of the service list mapping is incorrect or the service list specified by SERVL is less than the minimum requirement. If the version is less than 2, please recompile. 02 In a current RECEIVE request, the previous request was not one of the following: AQUERY, CMOUNT, CQUERY, DQUERY, EJECT, IQUERY, LQUERY, MINVENT, MOUNT, RELEASE (VGS only), SETVCAT, SQUERY; or the previous return code was not zero. 03 Phase IJJTLIB is not loaded in the SVA. 04 Either the previous request was LBSERV CANCEL and there is no more information available to receive, or the previous function has already completed.

Figure 27 (Page 2 of 2). Reason Codes Generated by VSE

Reason	Meaning	Explanation/Action	
63xx	LBSERV failure	05	User submitted a RECEIVE request but the main ECB has not been posted. Issue a WAIT macro on the main ECB before submitting the RECEIVE request.
		06	A MOUNT request is already pending for this CUU; try later.
		07	There is no more to RELEASE for this CUU.
		08	An invalid (out-of-partition) address has been specified for the ECB.
		09	An invalid (out-of-partition) address has been specified for LIBNAME.
		10	An invalid (out-of-partition) address has been specified for VOLSER.
		11	An invalid (out-of-partition) address has been specified for SRCCAT.
		12	A RELEASE request was given for a device not being ready, a rewind unload may have been already performed. Continue anyway with a RECEIVE request.
		13	An invalid CUU was specified for a MOUNT request.
		14	The specified CUU in a MOUNT request is in use, CUU is in ready state.
6400	GETVIS failed	There is no more storage available in the GETVIS area. Please check the return code (which is the return code coming from the GETVIS macro).	
6500	MODFLD failed	An error in the MODFLD macro occurred. Please check the return code (which is the return code coming from the MODFLD macro).	

LFCB (Load Forms Control Buffer) Macro



Requirements for the caller:

AMODE: 24
RMODE: 24
ASC Mode: Primary

The macro can be used to load the forms control buffer (FCB) of a printer dynamically. This printer must not be an IBM 3800; the macro is ignored for an IBM 3800. An FCB whose contents have been changed by means of this macro retains the changed contents until one of the following occurs:

- another LFCB macro is issued for the printer
- an LFCB command is issued for the printer
- the SYSBUFLD program is executed to reload the printer's FCB.
- IPL is performed for the system.

The macro, when executed, generates messages to request operator action (such as changing forms), whenever manual action is required. It informs the operator that the FCB of the specified printer has been reloaded.

If the FCB image to be loaded is to control horizontal copying on an IBM 4248, processing the macro turns on the function. Horizontal copying is turned off again when one of the following occurs:

- Your program issues a CNTRL macro that specifies ORDER,DHC.
- Your program issues a CLOSE (or CLOSER) macro for the printer file.
- The system initiates an automatic close for the printer file at the end of the job step.
- The logical unit SYSLST is assigned to the printer for which your program issues the LFCB macro.

SYSxxx
The name of the logical unit associated with the printer whose FCB is to be loaded. You can specify one of the following:

SYSLST

SYSLOG If assigned to the printer; the results of the initiated FCB load operation are unpredictable.

SYSnnn A programmer logical unit assigned to a printer owned by the partition in which the program runs.

phasename
The name by which the phase containing the applicable FCB image is cataloged in the accessible sublibrary. For information on the contents and format of an FCB image, see “Chapter 7. System Buffer Load (SYSBUFLD)” in the *VSE/ESA System Control Statements*.

NULMSG
This operand specifies that the 80-character verification message, which is normally printed following the FCB load operation, is to be suppressed. This operand, if given, causes the system to continue normal processing immediately after the FCB load operation has been completed. The operator cannot verify that the proper forms are placed on the printer.

If you omit the operand, the system:

1. Prints the last 80 characters of the phase identified by phasename.
2. Positions the printer to the first printable line on the forms.

FORMS=xxxx
This operand specifies the type of forms to be used on the printer whose FCB is being reloaded. For xxxx, a string of up to four alphameric characters can be specified. The specified form number is included in a message to the operator.

LPI=n The operand specifies the desired number of lines per inch. For n, you can specify either 6 or 8.

Do not specify this operand if the number of lines per inch is controlled by the FCB. If you code this operand for such a printer and the specified number disagrees with the lines-per-inch setting in the new buffer image, the system does not perform the FCB load operation.

When issued for a non-FCB controlled printer, the macro causes the operand to be included in a message to the operator.

Return Codes in Register 15

Successful completion of the FCB load operation is indicated to your program by a return code of 0. If the operation fails, register 15 contains one of the return codes listed below; in this case the FCB retains its original contents.

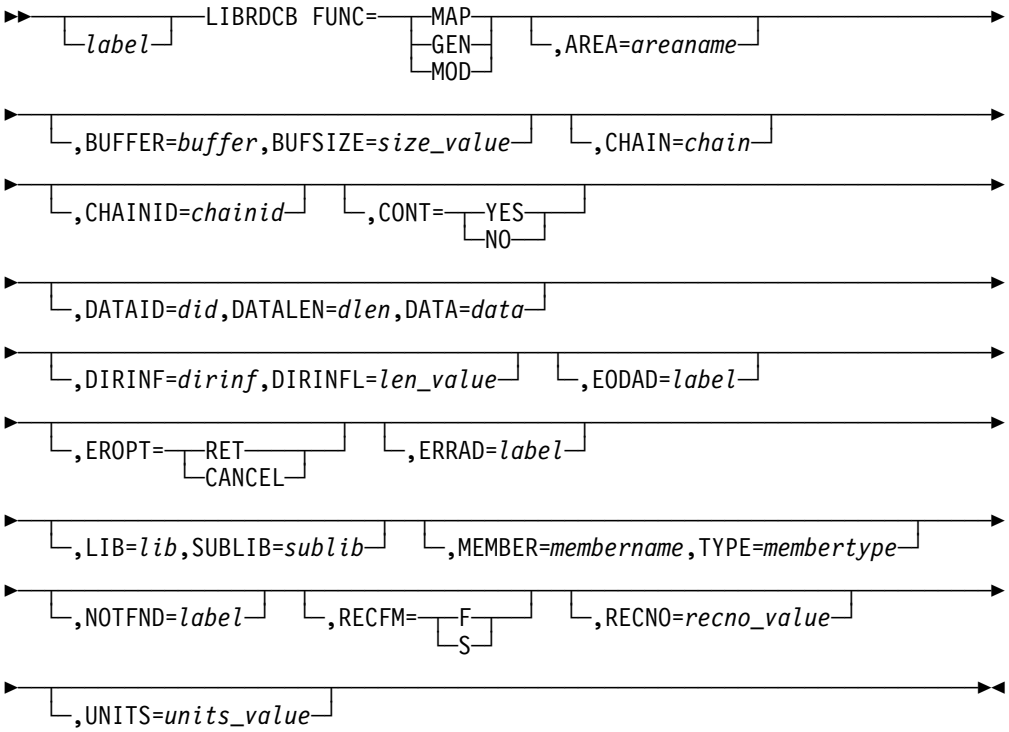
Note: For an IBM 3800, register 15 contains 0, although the macro was not executed.

The return codes and their meanings are:

X'04' For the printer, the number of lines per inch is controlled by the FCB. The LPI operand specified in the macro disagrees with lines-per-inch setting in the FCB image.
X'08' No LUB is available for the specified logical unit.
X'0C' The specified logical unit has not been assigned or is assigned IGN (ignore), or it is currently unassigned.
X'10' The specified logical unit is assigned to a device without an FCB.
X'14' The printer assigned to the specified logical unit is down.
X'18' The specified FCB image phase has not been found.
X'1C' The specified FCB image phase for the printer assigned to the specified logical unit is invalid or has an incorrect length or incorrect index byte, or the FCB data is out of range, or channel 1 is missing.

By testing register 15, you can determine in your program whether or not the operation has failed. If the operation has failed, you can either terminate the job step or continue processing. Should you decide to continue processing, then the system bypasses the execution of the LFCB macro.

LIBRDCB (Librarian Data Control Block) Macro



Requirements for the caller:

AMODE: 24

RMODE: 24

ASC Mode: Primary

This macro causes the Librarian Data Control Block (LDCB) for the Library Access Service either to be generated, or to be modified, or to be mapped. If it is generated, the addressed area will be cleared and all specified operand values are inserted. Operand specifications which are given with the function macro LIBRM will overwrite these initial specifications (always the latest specification of an operand will be valid).

FUNC Indicates the main function of the macro LIBRDCB.

MAP
Provides a mapping DSECT of the LDCB. No other operands are allowed.

GEN
Generates the LDCB at the given address.

MOD
Modifies the LDCB at the given address.

AREA=areaname
Specifies the name of the area where the LDCB is to be generated or modified.

All other operands are described together with the macros to which they apply.

Notes:

1. The LDCB must be generated (with FUNC=GEN) to be used as control block before any LIBRM request can be given.
2. The DSECT of the LDCB is created automatically, if necessary, for FUNC=GEN|MOD or any LIBRM request.
3. The registers used in expanded code are 0 and 1.

Library Macro Notation

- Macro operands specified with lowercase characters are either data areas or numeric values:
  - Each possible RX-type address can be used to address a data area. If register notation is used (registers 2 - 12), the specified register is expected to contain the address of the area.
  - Numeric values (for operands like BUFSIZE) can be specified in register notation or as a self-defining term. If register notation is used, the specified register is expected to contain the numeric value.
- Operands which can be specified in more than one macro, but **must** eventually be specified somewhere, are also shown as optional. For example, BUFFER may be specified in the LIBRDCB macro or at OPEN time, but if not done there, it must be specified in a GET or PUT request. If it is, for instance, specified in both the LIBRDCB and the GET macro, then the latest specification will be taken.

The following figure shows the possible combinations of macros and operands.

Operand	Macro						
				STATE			
	LIBRDCB	LIBDEF	LIBDROP	LIB	SUBLIB	MEMBER	CHAIN
BUFFER, BUFSIZE	o	-	-	-	-	-	-
CATALOG	-	-	-	-	-	-	O
CHAIN	o	m	-	-	-	-	m
CHAINID	o	m	m	-	-	m	m
CONT	o	-	-	-	o	o	-
DATA, DATAID,	o	-	-	-	-	o	-
DATALEN	o	-	-	-	-	o	-
DIRINF, DIRINFL	o	-	-	o	o	o	-
DIRNO	-	-	-	-	O	O	-
EODAD	o	-	-	-	-	-	-
EROPT	o	o	o	o	o	o	o
ERRAD	o	o	o	o	o	o	o
LIB, SUBLIB	o	-	-	m	m	m	-
LOCKID	-	-	-	-	-	O	-
MEMBER, TYPE	o	-	-	-	-	m	-
MOVELEN	-	-	-	-	-	-	-
NMEMBER, NTYPE	-	-	-	-	-	-	-
NOTECTL	-	-	-	-	-	-	-
NOTEINF	-	-	-	-	-	-	-
NOTFND	o	-	-	o	o	o	o
RECFM	o	-	-	-	-	-	-
RECNO	o	-	-	-	-	-	-
SCOPE	-	-	-	-	-	-	O
SYSIPT	-	-	-	-	-	-	-
TYPEFLE	-	-	-	-	-	-	-
UNITS	o	-	-	-	-	-	-
COMMIT	-	-	-	-	-	-	-

Abbreviations:

o Operand can be specified in a previous LIBRM or LIBRDCB request; if it is not specified here, a default value will be taken.

O Operand is optional and can be specified only in the indicated LIBRM macro.

m Operand can be specified in a previous LIBRM or LIBRDCB request, but if this has not been done, it must be specified here.

M Operand is mandatory and must be specified only in the indicated LIBRM macro.

Figure 28 (Part 1 of 2). Operand Notation for LIBRM Requests

Operand	Macro								
	RENAME	DELETE	OPEN	GET	PUT	NOTE	POINT	CLOSE	(UN)LOCK
BUFFER, BUFSIZE	-	-	o	m	m	-	-	-	-
CATALOG	-	-	-	-	-	-	-	-	-
CHAIN	-	-	-	-	-	-	-	-	-
CHAINID	m	m	m	-	-	-	-	-	-
COMMIT	-	-	-	-	-	-	-	O	-
CONT	-	-	-	-	-	-	-	-	-
DATA, DATAID,	-	-	o	-	-	-	-	-	-
DATALEN	-	-	o	-	-	-	-	-	-
DIRINF, DIRINFL	-	-	o	-	-	-	-	-	-
DIRNO	-	-	-	-	-	-	-	-	-
EODAD	-	-	-	o	-	-	-	-	-
EROPT	o	o	o	o	o	o	o	o	-
ERRAD	o	o	o	o	o	o	o	o	-
LIB, SUBLIB	m	m	m	-	-	-	-	-	-
LOCKID	-	-	-	-	-	-	-	-	M
MEMBER, TYPE	m	m	m	-	-	-	-	-	-
MOVELEN	-	-	-	O	-	-	-	-	-
NMEMBER, NTYPE	M	-	-	-	-	-	-	-	-
NOTECTL	-	-	-	-	-	O	O	-	-
NOTEINF	-	-	-	-	-	O	O	-	-
NOTFND	o	-	o	-	-	-	-	-	-
RECFM	-	-	o	-	-	-	-	-	-
RECNO	-	-	-	o	o	-	-	-	-
SCOPE	-	-	-	-	-	-	-	-	-
SYSIPT	-	-	O	-	-	-	-	-	-
TYPEFLE	-	-	O	-	-	-	-	-	-
UNITS	-	-	-	o	o	-	-	-	-

Abbreviations:

o Operand can be specified in a previous LIBRM or LIBRDCB request; if it is not specified here, a default value will be taken.

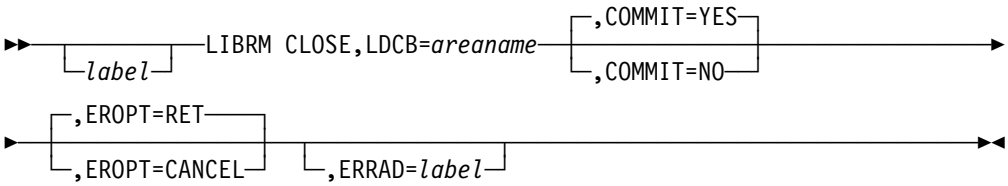
O Operand is optional and can be specified only in the indicated LIBRM macro.

m Operand can be specified in a previous LIBRM or LIBRDCB request, but if this has not been done, it must be specified here.

M Operand is mandatory and must be specified only in the indicated LIBRM macro.

Figure 28 (Part 2 of 2). Operand Notation for LIBRM Requests

LIBRM CLOSE (Close Library Member) Macro



Requirements for the caller:

AMODE: 24

RMODE: 24

ASC Mode: Primary

This macro closes the member addressed by the LDCB, so that no further GET/PUT or NOTE/POINT access is possible.

The directory entry for a member opened for OUTPUT or INOUT is written to the sublibrary.

For a member opened for OUTPUT or INOUT it is possible to prevent (decommit) its cataloging into the directory by specifying COMMIT=NO. The original member (if any) will stay in the sublibrary.

Empty members (members opened for OUTPUT or INOUT without a PUT request) are not cataloged.

For members opened for INOUT, the directory entry is written with the member name and member type which is addressed in the LDCB at CLOSE time. Thus, it is possible to catalog the new member version under a different name while keeping the old version of the member in the sublibrary.

CLOSE clears member-related data like RECFM, RECNO, UNITS, and the options for the OPEN and CLOSE request in the LDCB and frees the resources (virtual storage space, locks, etc.) that OPEN used to construct and protect library control blocks and library objects.

See also “Library Macro Notation” on page 253 for details on register notation and possible operand specifications.

LDCB=areaname
Specifies the address of the LDCB (Librarian Data Control Block) for the request.

COMMIT=YES | NO
Is applicable only for a member opened for OUTPUT or INOUT:

YES the member is cataloged (default).

NO the member is not cataloged.

EROPT=RET | CANCEL
Defines an error handling option which will be taken if the function cannot be performed (return code > 12).

RET
Processing will be continued, either by a normal return or by branching to the ERRAD exit.

CANCEL
Processing will be canceled. A librarian error message will be issued to SYSLOG.

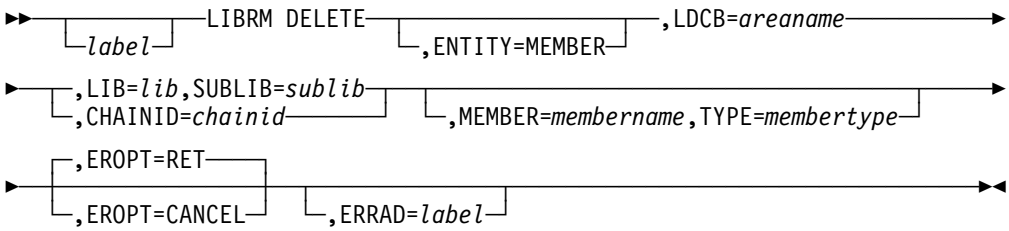
ERRAD=label
Specifies a label to which the Librarian will branch if the function cannot be performed because of an error (return code > 12).

Return Codes

Return Code	Reason Code	Meaning
0	0	Member has been closed.
12	12	Library is full (for OUTPUT, INOUT). New member has been purged, old kept.
16	xx	External system error with feedback code and message.
20	xx	Internal system error with feedback code and message.
32	0	Access control failed (with message L163I).

The external system error feedback codes are described in Appendix D, “Librarian Feedback Codes” on page 455. All other (internal) feedback codes are described in *VSE Central Functions Librarian Diagnosis Reference*, SC33-6330.

LIBRM DELETE (Delete Library Member) Macro



Requirements for the caller:

AMODE: 24

RMODE: 24

ASC Mode: Primary

The DELETE (MEMBER) function is used to delete a library member. The member can be specified fully qualified, or it can be searched in a sublibrary chain.

See also “Library Macro Notation” on page 253 for details on register notation and possible operand specifications.

LDCB=areaname
Specifies the address of the LDCB (Librarian Data Control Block) for the request.

LIB=lib
Specifies the address of an area where the library name (1 to 7 alphameric characters) is stored.

SUBLIB=sublib
Specifies the address of an area where the sublibrary name (1 to 8 alphameric characters) is stored.

CHAINID=chainid
Specifies the address of an area where the sublibrary chain identifier (1 to 8 alphameric characters) is stored.

MEMBER=membername
Specifies the address of an area where the member name (1 to 8 alphameric characters) is stored.

TYPE=membertype
Specifies the address of an area where the member type (1 to 8 alphameric characters) is stored.

EROPT=RET | CANCEL
Defines an error handling option which will be taken if the function cannot be performed (return code > 12).

RET
Processing will be continued, either by a normal return or by branching to the ERRAD exit.

CANCEL
Processing will be canceled. A librarian error message will be issued to SYSLOG.

ERRAD
Specifies a label to which the Librarian will branch if the above function cannot be performed because of an error (return code > 12).

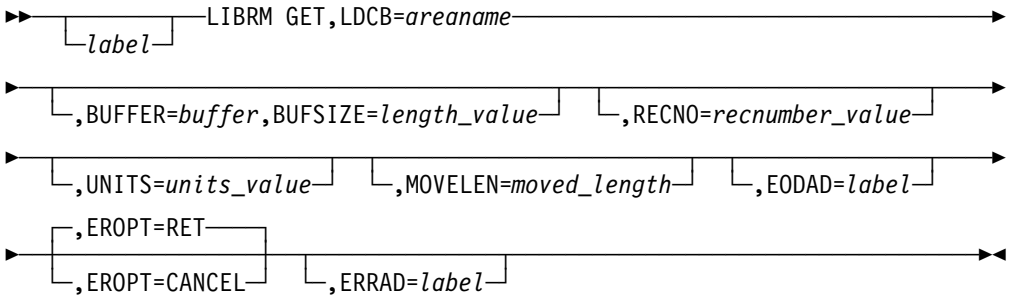
Return Codes

Return Code	Reason Code	Meaning
0	0	Member has been deleted.
4	0	The specified member does not exist.
12	0	The specified sublibrary does not exist.
	4	The specified library does not exist.
	8	The specified chain does not exist.
16	xx	External system error with feedback code and message.
20	xx	Internal system error with feedback code and message.
32	0	Access control failed (with message L163I).

The external system error feedback codes are described in Appendix D, “Librarian Feedback Codes” on page 455. All other (internal) feedback codes are described in *VSE Central Functions Librarian Diagnosis Reference*, SC33-6330.

Note: If both LIB/SUBLIB and CHAINID are specified, LIB/SUBLIB will be taken and CHAINID is ignored.

LIBRM GET (Get Library Member) Macro



Requirements for the caller:

AMODE: 24

RMODE: 24

ASC Mode: Primary

This macro retrieves one or more records of a member with record format F or a string of bytes of a member with record type S in the user's work area specified by BUFFER and BUFSIZE.

The starting position of the retrieval within the member is always the current position. This position can be altered by specifying the starting record number (for record format F) or the starting byte number (for record format S) in RECNO.

The amount of data to be returned is controlled by the UNITS operand: For record format S specify the number of bytes to be read, for record format F the number of records to be retrieved with the request. You can either retrieve one or more records or bytes (UNITS>0), or fill up the whole buffer with member data (UNITS=0). The length of the returned data can be obtained via the MOVELEN operand.

The current position is updated either to the record (for record format F) or byte (for record format S) following the last returned member data.

See also “Library Macro Notation” on page 253 for details on register notation and possible operand specifications.

LDCB=areaname
Specifies the address of the LDCB (Librarian Data Control Block) for the request.

BUFFER=buffer
Specifies the address of the caller's work area where the member data is to be returned.

BUFSIZE=length-value
Specifies the length (as a numeric value) of the caller's work area.

RECNO=recnumber-value
Specifies the starting point (as a numeric value) of the data retrieval within the member. The value indicates the offset either in number of records (for

record format F) or in number of bytes (for format S) relative to the start of the member.

After the GET request, RECNO is set to its default value, which is the current member position.

UNITS=units-value
Specifies, as a numeric value, the number of records or bytes to be retrieved.

If UNITS=0, the buffer is filled up with member data, either until the buffer is full or the member is exhausted. If, for record format 'F' the buffer cannot contain all records, the last record will not be truncated (except when the last record is also the first record).

The default is 1 for record format F, and 0 for record format S. This default will be taken if RECFM is specified with the OPEN or LIBRDCB macro, or if UNITS is not specified at all.

MOVELEN=moved-length
Specifies the address of a fullword where the length of the data written into the caller's work area is returned.

EODAD=label
Specifies a label to which the service will branch if the function is not performed because of an end-of-member condition (return code 8).

EROPT=RET | CANCEL
Defines an error handling option which will be taken if the function cannot be performed (return code > 12).

RET
Processing will be continued, either by a normal return or by branching to the ERRAD exit.

CANCEL
Processing will be canceled. A librarian error message will be issued to SYSLOG.

ERRAD
Specifies a label to which the Librarian will branch if the above function cannot be performed because of an error (return code > 12).

Return Codes

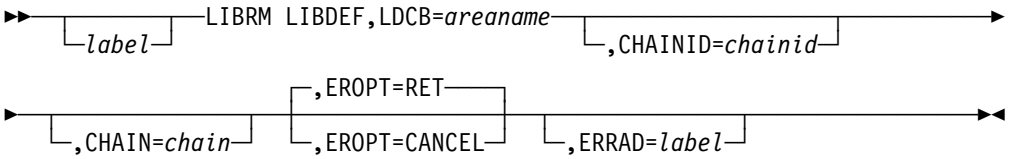
Return Code	Reason Code	Meaning
0	0	Record(s) or byte(s) returned (more available)
	4	For record format F: Record(s) returned, but work area is too small to return the requested amount of records (UNITS * LRECL > BUFSIZE). The last record in the work area may be truncated. For record format S: Byte(s) returned, but work area is too small to return the requested amount of bytes (UNITS > BUFSIZE).
4	0	Last record or byte returned.
	4	For record format F: Last record returned, but work area is too small to return the requested amount of records. The last record is truncated. For record format S: Not applicable.
	8	Last record or byte returned, but more records or bytes were requested than are available.
8	0	Reading past end-of-member (EOF).
16	xx	External system error with feedback code and message.
20	xx	Internal system error with feedback code and message.

The external system error feedback codes are described in Appendix D, “Librarian Feedback Codes” on page 455. All other (internal) feedback codes are described in *VSE Central Functions Librarian Diagnosis Reference*, SC33-6330.

Notes:

1. For a GET request with RECFM=F and UNITS=0, the last record will only be truncated if the size of the work area (BUFSIZE) is not large enough to hold at least one record. Reason code 4 will be returned in that case.
2. If a GET request is issued with a RECNO value outside the member range, return code 8 is issued.
3. A GET request for an empty member yields return code 8 and reason code 0.
4. The member will be internally closed for all exceptions with the return code > 8.

LIBRM LIBDEF (Define Sublibrary Chain) Macro



Requirements for the caller:

AMODE: 24

RMODE: 24

ASC Mode: Primary

This macro causes a chain of up to 15 sublibraries to be built and identified by the name given with CHAINID. Referring to the name of the chain, you can look up a member by searching the chain sequentially (for example, for LIBRM STATE or LIBRM OPEN).

The chain is valid for the duration of a task. Only the task that established the chain can access it. Other tasks from the same or another partition cannot access the chain. However, those tasks can establish a chain with the same name.

See also “Library Macro Notation” on page 253 for details on register notation and possible operand specifications.

LDCB=areaname
Specifies the address of the LDCB (Librarian Data Control Block) for the request.

CHAINID=chainid
Specifies the address of an area where the sublibrary chain identifier (1 to 8 alphameric characters) is stored.

CHAIN=chain
Specifies the address of an array of 1 to 15 entries with DS CL7 containing the library name DS CL8 containing the sublibrary name.

The last entry must be followed by 4 characters 'FFFFFFFF'X (decimal value: -1). They do not require fullword boundary.

EROPT=RET | CANCEL
Defines an error handling option which will be taken if the function cannot be performed (return code > 12).

RET
Processing will be continued, either by a normal return or by branching to the ERRAD exit.

CANCEL
Processing will be canceled. A librarian error message will be issued to SYSLOG.

ERRAD=label
Specifies a label to which the Librarian will branch if the above function cannot be performed because of an error (return code > 12).

Return Codes

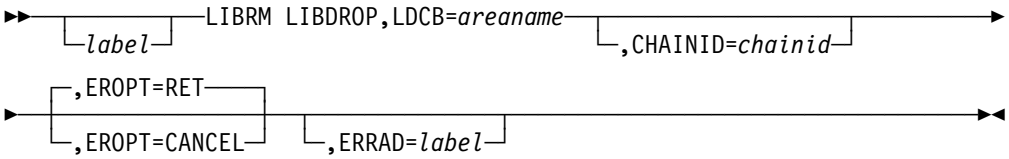
Return Code	Reason Code	Meaning
0	0	Chain established.
8		Chain not established, since more than 15 sublibraries were specified.
12		Chain not established.
	0	Chain contains non-existing sublibrary.
	4	Chain contains non-existing library.
16	xx	External system error with feedback code and message.
20	xx	Internal system error with feedback code and message.
32	0	Access control failed (with message L163I).

The external system error feedback codes are described in Appendix D, “Librarian Feedback Codes” on page 455. All other (internal) feedback codes are described in *VSE Central Functions Librarian Diagnosis Reference*, SC33-6330.

Notes:

1. The chain exists as long as it is not overwritten (with the same CHAINID) or dropped (with the LIBDROP macro), or until the task is ended.
2. A LIBDEF with CHAINID=ACCESS|SEARCH|CATALOG corresponds to the sublibrary specifications on a Librarian ACCESS and CONNECT (SEARCH and CATALOG) command, respectively. They will overwrite each other if used within the same task.

LIBRM LIBDROP (Drop Sublibrary Chain) Macro



Requirements for the caller:

AMODE: 24

RMODE: 24

ASC Mode: Primary

This macro causes the sublibrary chain referenced by CHAINID to be dropped.

See also “Library Macro Notation” on page 253 for details on register notation and possible operand specifications.

LDCB=areaname
Specifies the address of the LDCB (Librarian Data Control Block) for the request.

CHAINID=chainid
Specifies the address of an area where the sublibrary chain identifier (1 to 8 alphameric characters) is stored.

EROPT=RET | CANCEL
Defines an error handling option which will be taken if the function cannot be performed (return code > 12).

RET
Processing will be continued, either by a normal return or by branching to the ERRAD exit.

CANCEL
Processing will be canceled. A librarian error message will be issued to SYSLOG.

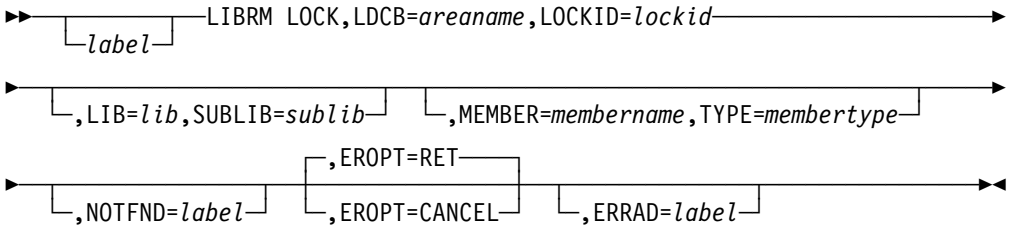
ERRAD=label
Specifies a label to which the Librarian will branch if the above function cannot be performed because of an error (return code > 12).

Return Codes

Return Code	Reason Code	Meaning
0	0	Chain dropped.
4	0	Chain does not exist.
16	xx	External system error with feedback code and message.
20	xx	Internal system error with feedback code and message.

The external system error feedback codes are described in Appendix D, “Librarian Feedback Codes” on page 455. All other (internal) feedback codes are described in *VSE Central Functions Librarian Diagnosis Reference*, SC33-6330.

LIBRM LOCK (Lock Library Member) Macro



Requirements for the caller:

AMODE: 24

RMODE: 24

ASC Mode: Primary

The LOCK (MEMBER) function is used to lock a library member for any write or update access. The member must be fully qualified. The member can be unlocked again with the LIBRM UNLOCK macro.

See also “Library Macro Notation” on page 253 for details on register notation and possible operand specifications.

LDCB=areaname
Specifies the address of the LDCB (Librarian Data Control Block) for the request.

LOCKID=lockid
Specifies a lock word by which the lock operation is uniquely identified. The **lockid** is a string of up to eight alphanumeric characters; it may not be generic. The specified lockid must be used when the library member is to be unlocked again (with the corresponding UNLOCK function).

The **lockid** in the LDCB will be cleared after processing.

LIB=lib
Specifies the address of an area where the library name (1 to 7 alphameric characters) is stored.

SUBLIB=sublib
Specifies the address of an area where the sublibrary name (1 to 8 alphameric characters) is stored.

MEMBER=membername
Specifies the address of an area where the member name (1 to 8 alphameric characters) is stored. No generic specification is allowed.

TYPE=membertype
Specifies the address of an area where the member type (1 to 8 alphameric characters) is stored.

NOTFND=label
Specifies a label to which the service will branch if the specified member does not exist (return code 8 for members opened for INPUT), or if the

specified library or sublibrary does not exist or is not accessible (return code 12).

EROPT=RET | CANCEL
Defines an error handling option which will be taken if the function cannot be performed (return code > 12).

RET
Processing will be continued, either by a normal return or by branching to the ERRAD exit.

CANCEL
Processing will be canceled. A librarian error message will be issued to SYSLOG.

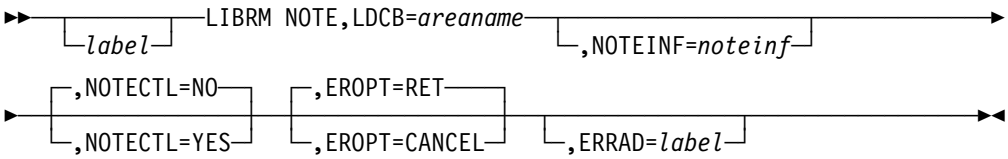
ERRAD
Specifies a label to which the Librarian will branch if the above function cannot be performed because of an error (return code > 12).

Return Codes

Return Code	Reason Code	Meaning
0	0	Member has been locked.
0	8	The macro is ignored, no member will be locked. Option IGNLOCK is on.
4	0	The specified member was already locked.
8	0	The specified member does not exist.
	4	The specified member is locked with a different lockid.
12	0	The specified sublibrary does not exist.
	4	The specified library does not exist.
16	xx	External system error with feedback code and message.
20	xx	Internal system error with feedback code and message.
32	0	Access control failed (with message L163I).

The external system error feedback codes are described in Appendix D, “Librarian Feedback Codes” on page 455. All other (internal) feedback codes are described in *VSE Central Functions Librarian Diagnosis Reference*, SC33-6330.

LIBRM NOTE (Note Member Address) Macro



Requirements for the caller:

AMODE: 24

RMODE: 24

ASC Mode: Primary

This macro saves the current member position, that is, the position following the record or byte that was last read from or written to the library member specified in the LDCB.

The member position for members opened for INPUT and INOUT is the GET position, for members opened for OUTPUT it is the PUT position.

For NOTECTL=NO the Librarian keeps track of the noted position by itself in a last-in first-out (LIFO) mechanism, NOTE pushing an element to the stack and POINT popping up the stack. The stack has a depth of 20 elements.

Specify NOTECTL=YES if you want to control this information yourself. In this case, a 32-byte area NOTEINF has to be provided which obtains the member's position.

It is not possible to switch from NOTECTL=NO to NOTECTL=YES and vice versa.

See also “Library Macro Notation” on page 253 for details on register notation and possible operand specifications.

LDCB=areaname
Specifies the address of the LDCB (Librarian Data Control Block) for the request.

NOTEINF=noteinf
Specifies the address of a 32-byte area in which the NOTE information is to be returned. This operand is required if NOTECTL=YES has been specified.

NOTECTL=NO | YES
Specifies who is to maintain the NOTE information:

NO
The Librarian keeps track of the NOTE information (LIFO mechanism).

YES
The caller maintains the information himself. In this case, the NOTEINF operand is required.

EROPT=RET | CANCEL
Defines an error handling option which will be taken if the function cannot be performed (return code > 12).

RET
Processing will be continued, either by a normal return or by branching to the ERRAD exit.

CANCEL
Processing will be canceled. A librarian error message will be issued to SYSLOG.

ERRAD=label
Specifies a label to which the Librarian will branch if the function cannot be performed because of an error (return code > 12).

Return Codes

Return Code	Reason Code	Meaning
0	0	NOTE information has been extracted.
16	xx	External system error with feedback code and message.
20	xx	Internal system error with feedback code and message.

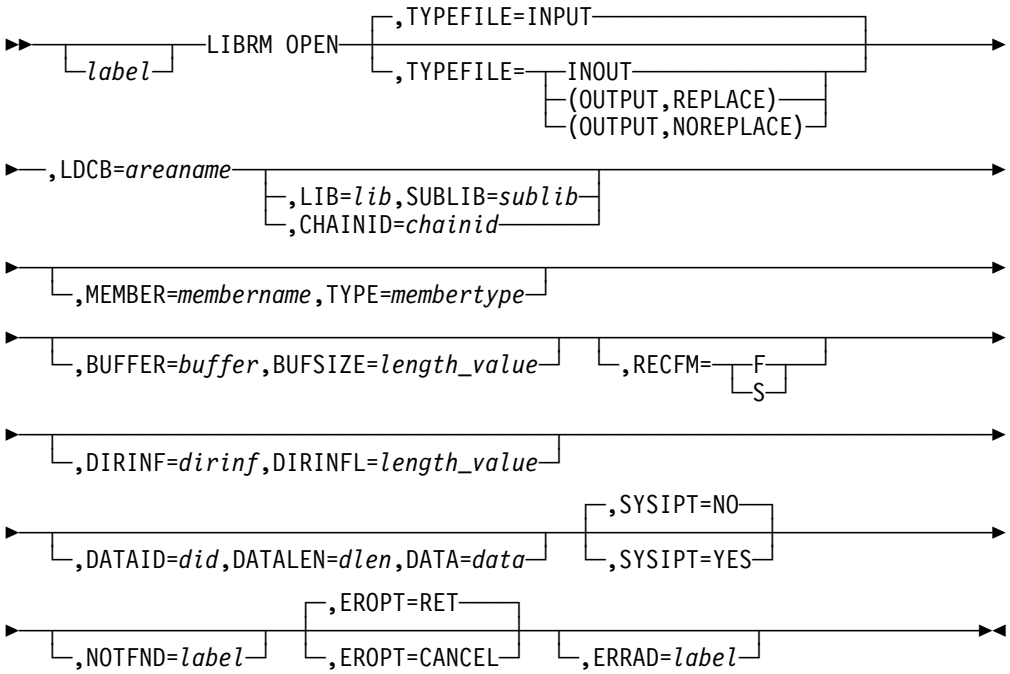
The external system error feedback codes are described in Appendix D, “Librarian Feedback Codes” on page 455. All other (internal) feedback codes are described in *VSE Central Functions Librarian Diagnosis Reference*, SC33-6330.

Notes:

1. If a stack overflow occurs (that is, if the number of NOTE requests minus the number of POINT requests is greater than 20), return code 16 is passed back with a feedback code.
2. A member will automatically be closed if the return code is higher than 12.

Please note that instead of specifying LIBRM NOTE also LIBRM NOTEF can be specified. This must be done when the PL/X 1.4 (or higher) compiler is used.

LIBRM OPEN (Open Library Member) Macro



Requirements for the caller:

AMODE: 24

RMODE: 24

ASC Mode: Primary

This macro opens the member defined by MEMBER and TYPE in the sublibrary LIB/SUBLIB, if given, otherwise in the chain identified by CHAINID. The member is opened either for input (INPUT), or update (INOUT), or output (OUTPUT).

See also “Library Macro Notation” on page 253 for details on register notation and possible operand specifications.

TYPEFLE=INPUT

The first occurrence of the member in the chain sequence is opened for INPUT. The current (GET) position is at the start of the member. No PUT requests are accepted.

For INPUT (or INOUT), directory information will be provided in the area specified by DIRINF (control block layout INLCMBST). User-provided directory information identified by DATAID is returned in DATA, its length in DATALEN.

TYPEFLE=INOUT

The first occurrence of the member in the chain sequence is opened for INOUT. If the member does not exist, a new member is opened in the first sublibrary of the chain. GET position is at the start of the existing member, PUT position is at the start of the new copy of the member.

TYPEFLE=(OUTPUT,REPLACE)
The member is opened for OUTPUT in the first sublibrary of the chain.

TYPEFLE=(OUTPUT,NOREPLACE)
The member is only opened for OUTPUT if it does not exist in the first sublibrary of the chain.

For OUTPUT, no GET or POINT requests are allowed. The presence of SYSIPT data is indicated by the SYSIPT operand.

LDCB=areaname
Specifies the address of the LDCB (Librarian Data Control Block) for the request.

LIB=lib
Specifies the address of an area where the library name (1 to 7 alphameric characters) is stored.

SUBLIB=sublib
Specifies the address of an area where the sublibrary name (1 to 8 alphameric characters) is stored.

CHAINID=chainid
Specifies the address of an area where the sublibrary chain identifier (1 to 8 alphameric characters) is stored.

MEMBER=membername
Specifies the address of an area where the member name (1 to 8 alphameric characters) is stored.

TYPE=membertype
Specifies the address of an area where the member type (1 to 8 alphameric characters) is stored.

BUFFER=buffer
Specifies the address of the caller's work area where a GET request returns the input or a PUT request gets the output.

BUFSIZE=length-value
Specifies the length (as a numeric value) of the caller's work area.

RECFM=F | S
Specifies the logical record format:

F specifies record format FIXED,

S specifies record format STRING.

If TYPEFLE=INPUT or INOUT and the member exists, RECFM is checked. If TYPEFLE=OUTPUT or INOUT and the member does not exist, F is assumed as default.

For TYPE=DUMP or PHASE, record format F is not allowed; S is assumed as default.

For TYPE=PROC, OBJ, or source program, record format S is not allowed.

DIRINF=dirinf
Specifies the address of an area where member directory information is returned.

DIRINFL=length-value
Specifies the length (numeric value) of DIRINF area.

DATAID=did
Specifies the address of an area where the 1 to 4 character alphameric identifier for user-provided directory information is specified.

DATALEN=dlen
Specifies the address of a fullword where the length of the DATA area is given. The actual length of the user-provided information will be returned by the service.

DATA=data
Specifies the address of the area where the user-provided directory information is returned. The area specified must have a minimal length of DATALEN.

SYSIPT=NO | YES
Indicates whether the member contains SYSIPT data:

NO
No SYSIPT data in the member.

YES
The member contains SYSIPT data.

SYSIPT is only possible for OPEN(OUTPUT). Default is NO.

NOTFND=label
Specifies a label to which the service will branch if the specified member does not exist (return code 8 for members opened for INPUT), or if the specified library, sublibrary or chain does not exist or is not accessible (return code 12).

EROPT=RET | CANCEL
Defines an error handling option which will be taken if the function cannot be performed (return code > 12).

RET
Processing will be continued, either by a normal return or by branching to the ERRAD exit.

CANCEL
Processing will be canceled. A librarian error message will be issued to SYSLOG.

ERRAD=label
Specifies a label to which the Librarian will branch if the above function cannot be performed because of an error (return code > 12).

Return Codes

Return Code	Reason Code	Meaning
0	0	OPEN(INPUT): Member exists and is opened for GET requests.
		OPEN(OUTPUT,REPLACE): Member exists and is opened for replacement by subsequent PUT requests.
		OPEN(OUTPUT,NOREPLACE): Member does not exist. It is opened for creation.
		OPEN(INOUT): Member exits and is opened for creating a new copy.
	4	OPEN(OUTPUT,REPLACE): Member does not exist. It is opened for creation.
4	0	OPEN(INOUT): Member does not exist. It is opened for creation.
8	0	OPEN(INPUT): Member does not exist. No open is done.
		OPEN(OUTPUT,NOREPLACE): Member already exists. It is not opened.
12	0	The specified sublibrary does not exist.
	4	The specified library does not exist.
	8	The specified chain does not exist.
16	xx	External system error with feedback code and message.
20	xx	Internal system error with feedback code and message.
32	0	Access control failed (with message L163I).

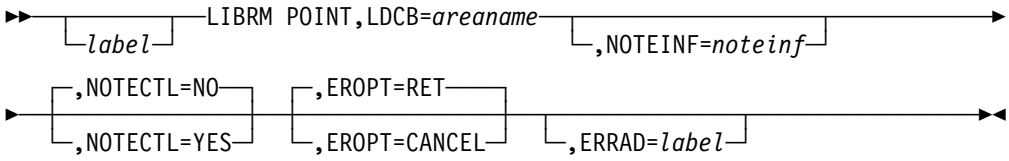
The external system error feedback codes are described in Appendix D, “Librarian Feedback Codes” on page 455. All other (internal) feedback codes are described in *VSE Central Functions Librarian Diagnosis Reference*, SC33-6330.

Notes:

1. If both LIB/SUBLIB and CHAINID are specified, LIB/SUBLIB will be taken and CHAINID will be ignored.
2. If RECFM has been specified, the GET/PUT UNITS value is set to its default.
3. The GET/PUT RECNO value, which defines the start position for the subsequent GET or PUT request, is initialized to the start of the member. It overwrites a previously specified value.
4. If it is not possible to return any user-provided directory information (operands DATA, DATALEN, DATAID), the fullword addressed with DATALEN will be set to zero. Return and reason codes are not affected.

   Processing will be ignored if DIRINF is not specified or its length is too small to return the member status.
5. Executable programs (TYPE=PHASE) can only be **read**.

LIBRM POINT (Point to Noted Member Record) Macro



Requirements for the caller:

AMODE: 24

RMODE: 24

ASC Mode: Primary

This macro repositions a member to the record identified by a previous NOTE macro.

POINT is not possible for OPEN(OUTPUT). For OPEN(INOUT), POINT refers to the old version of the member.

If you want to control the NOTE information yourself (NOTECTL=YES), you must provide the new position in an area specified by NOTEINF. In this case, the new position must be in the currently opened member. The NOTE information given with POINT must be obtained by a previous NOTE request and must not have been modified by the caller.

If you specify NOTECTL=NO, the Librarian takes the information of the most recent NOTE operation for member positioning. In this case, the number of POINT operations at any time must be less than or equal to the number of performed NOTE operations. If a member was opened by a LIB/SUBLIB specification, POINT is only possible for a member within the same sublibrary, otherwise a sublibrary chain specification has to be used.

After POINT, the number of requested units (UNITS operand of GET/PUT) is set to its default.

See also “Library Macro Notation” on page 253 for details on register notation and possible operand specifications.

LDCB=areaname
Specifies the address of the LDCB (Librarian Data Control Block) for the request.

NOTEINF=noteinf
Specifies the address of a 32-byte area in which the NOTE information is to be returned. This operand is required if NOTECTL=YES has been specified.

NOTECTL=NO | YES
Specifies who is to maintain the NOTE information:

NO
The Librarian keeps track of the NOTE information (LIFO mechanism).

YES
The caller maintains the information himself. In this case, the NOTEINF operand is required.

EROPT=RET | CANCEL
Defines an error handling option which will be taken if the function cannot be performed (return code > 12).

RET
Processing will be continued, either by a normal return or by branching to the ERRAD exit.

CANCEL
Processing will be canceled. A librarian error message will be issued to SYSLOG.

ERRAD=label
Specifies a label to which the Librarian will branch if the above function cannot be performed because of an error (return code > 12).

Return Codes

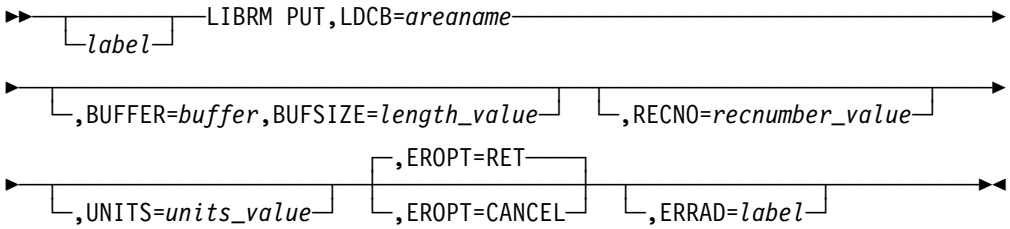
Return Code	Reason Code	Meaning
0	0	Position has been reset according to NOTE information.
12	28	POINT failed because of invalid NOTE information (old position kept).
16	xx	External system error with feedback code and message.
20	xx	Internal system error with feedback code and message.

The external system error feedback codes are described in Appendix D, “Librarian Feedback Codes” on page 455. All other (internal) feedback codes are described in *VSE Central Functions Librarian Diagnosis Reference*, SC33-6330.

Notes:

1. If a stack underflow occurs (that is, if the number of NOTE requests is smaller than the number of POINT requests), return code 16 is passed back with a feedback code.
2. The NOTE information is only valid between OPEN and CLOSE.
3. A member will automatically be closed if the return code is higher than 12.

LIBRM PUT (Put Library Member) Macro



Requirements for the caller:

AMODE: 24

RMODE: 24

ASC Mode: Primary

This macro writes one or more records of a member with record format F or a string of bytes of a member with record type S from the user's work area specified by BUFFER and BUFSIZE to the member specified in LDCB.

The starting position of the write operation within the member is always the current position. It can only be altered for members with record format S (via RECNO). For record format F, data can only be appended to the member. If, for record format S, the specified position is beyond the high end of the member, the member is extended with X'00' up to the specified position.

The amount of data to be written is controlled by the UNITS operand: It is possible to write one or more records or bytes (UNITS > 0), or to write the whole buffer contents to the member (UNITS=0).

For record format F, only complete records stored in the user's work area are written. A UNITS specification for one or more records (UNITS > 0) is internally limited by the size of the work area.

The current position is updated to the record or byte following the last written member data.

See also “Library Macro Notation” on page 253 for details on register notation and possible operand specifications.

LDCB=areaname
Specifies the address of the LDCB (Librarian Data Control Block) for the request.

BUFFER=buffer
Specifies the address of the caller's work area where the member data is provided.

BUFSIZE=length-value
Specifies the length of the caller's work area (only for UNITS=0).

RECNO=recnumber-value
Specifies the starting position (as a numeric value) in the member where the data is to be written.

Default is the member's current position. This default is active after each PUT. For record format F, data can only be appended to the current end of the member (RECNO specification is ignored).

UNITS=units-value
Specifies, as a numeric value, the number of records or bytes to be written.

If UNITS is 0, all data contained in the buffer is written to the member.

The default is 1 for record format F, and 0 for record format S. This default will be taken if RECFM is specified with the OPEN or LIBRDCB macro, or if UNITS is not specified at all.

EROPT=RET | CANCEL
Defines an error handling option which will be taken if the function cannot be performed (return code > 12).

RET
Processing will be continued, either by a normal return or by branching to the ERRAD exit.

CANCEL
Processing will be canceled. A librarian error message will be issued to SYSLOG.

ERRAD=label
Specifies a label to which the Librarian will branch if the above function cannot be performed because of an error (return code > 12).

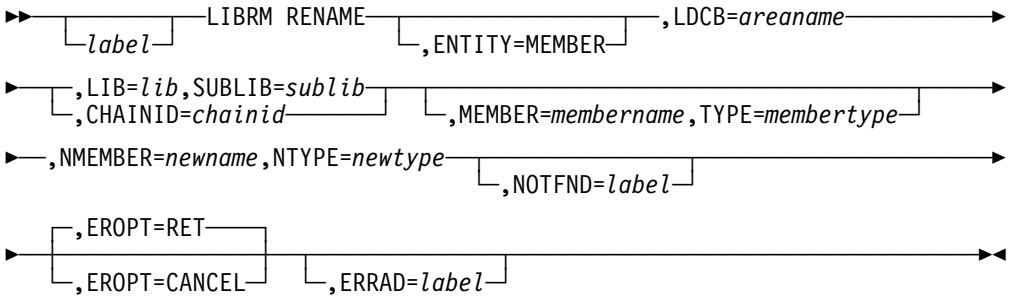
Return Codes

Return Code	Reason Code	Meaning
0	0	Record(s) byte(s) written to member.
8	0	For record format F: The user's work area does not contain a complete record. The request is ignored.
12	12	Library is full: Complete new member data is purged.
16	xx	External system error with feedback code and message.
20	xx	Internal system error with feedback code and message.

The external system error feedback codes are described in Appendix D, “Librarian Feedback Codes” on page 455. All other (internal) feedback codes are described in *VSE Central Functions Librarian Diagnosis Reference*, SC33-6330.

Note: The member will be automatically closed for all exceptions with return code > 8. The complete new member data is purged.

LIBRM RENAME (Rename Library Member) Macro



Requirements for the caller:

AMODE: 24

RMODE: 24

ASC Mode: Primary

The RENAME (MEMBER) function is used to rename a fully qualified member.

See also “Library Macro Notation” on page 253 for details on register notation and possible operand specifications.

LDCB=areaname
Specifies the address of the LDCB (Librarian Data Control Block) for the request.

LIB=lib
Specifies the address of an area where the library name (1 to 7 alphameric characters) is stored.

SUBLIB=sublib
Specifies the address of an area where the sublibrary name (1 to 8 alphameric characters) is stored.

CHAINID=chainid
Specifies the address of an area where the sublibrary chain identifier (1 to 8 alphameric characters) is stored.

MEMBER=membername
Specifies the address of an area where the **old** member name (1 to 8 alphameric characters) is stored.

TYPE=membertype
Specifies the address of an area where the **old** member type (1 to 8 alphameric characters) is stored.

NMEMBER=newname
Specifies the address of an area where the **new** member name (1 to 8 alphameric characters) is stored.

NTYPE=newtype
Specifies the address of an area where the **new** member type (1 to 8 alphameric characters) is stored.

NOTFND=label
Specifies the label to which the service will branch if the 'old' member name does not exist (return code 8, reason code 0), or if the specified library, sublibrary or chain does not exist (return code 12).

EROPT=RET | CANCEL
Defines an error handling option which will be taken if the function cannot be performed (return code > 12).

RET
Processing will be continued, either by a normal return or by branching to the ERRAD exit.

CANCEL
Processing will be canceled. A librarian error message will be issued to SYSLOG.

ERRAD=label
Specifies a label to which the Librarian will branch if the above function cannot be performed because of an error (return code > 12).

Return Codes

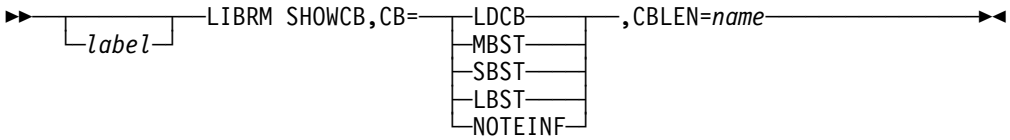
Return Code	Reason Code	Meaning
0	0	Member has been renamed.
4	0	Rename ignored: old and new member name and type are identical.
8	0	Old member does not exist.
	4	New member already exists.
	8	Record type (member-type) conflict.
12	0	The specified sublibrary does not exist.
	4	The specified library does not exist.
	8	The specified chain does not exist.
	12	The library is full.
16	xx	External system error with feedback code and message.
20	xx	Internal system error with feedback code and message.
32	0	Access control failed (with message L163I).

The external system error feedback codes are described in Appendix D, “Librarian Feedback Codes” on page 455. All other (internal) feedback codes are described in *VSE Central Functions Librarian Diagnosis Reference*, SC33-6330.

Notes:

1. If both LIB/SUBLIB and CHAINID are specified, LIB/SUBLIB will be taken and CHAINID will be ignored.
2. The NOTFND Exit will only be taken if the old member does not exist. If the new member already exists, return code 8 and reason code 4 will be returned.
3. For a member-TYPE conflict (for example, renaming a PHASE to a MACRO), return code 8 with reason code 8 is passed.
4. A library-full condition (return code 12, reason code 12) may be raised when renaming a member because of a directory block split.

LIBRM SHOWCB (Show Librarian Control Block) Macro



Requirements for the caller:

AMODE: 24

RMODE: 24

ASC Mode: Primary

The macro retrieves the length of the specified control block and returns it in the field CBLEN.

CB=LDCB
Identifies the LDCB (Librarian Data Control Block).

CB=MBST
Identifies the **member** status control block INLCMBST.

CB=SBST
Identifies the **sublibrary** status control block INLCSBST.

CB=LBST
Identifies the **library** status control block INLCLBST.

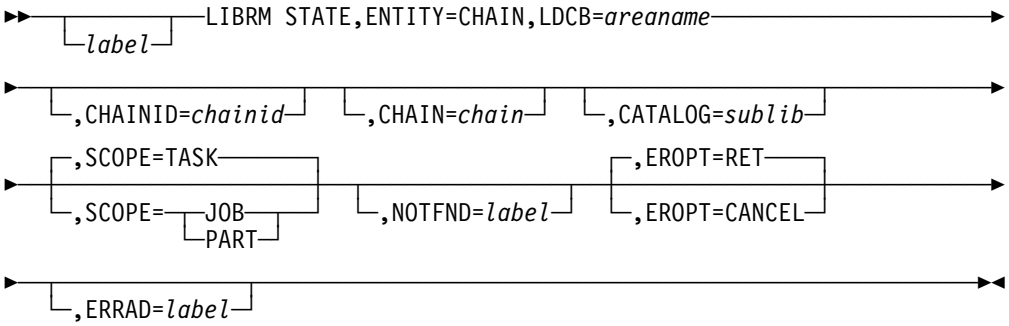
CB=NOTEINF
Identifies the **NOTE** information word.

CBLEN=name
Specifies the name of the fullword that is to receive the length of the identified control block.

Notes:

1. The SHOWCB macro service also requires registers 0, 1, 13, 14 and 15.
2. With the SHOWCB function the storage size for the used library control blocks can be determined at run time of the application. This gives the possibility to enlarge the control blocks without recompilation.
3. The CBLEN value returned for the LBST (library status) is the size necessary to hold all 16 extent information records.

LIBRM STATE CHAIN (Search Library Chain) Macro



Requirements for the caller:

AMODE: 24

RMODE: 24

ASC Mode: Primary

The STATE (CHAIN) function is used to determine whether a particular chain (identified by CHAINID) of sublibraries exists. If it exists, the sequence of sublibraries building up that chain is returned in the area specified by CHAIN.

With the CATALOG operand you can retrieve the name of the sublibrary specified in the external Job Control LIBDEF CATALOG statement.

See also “Library Macro Notation” on page 253 for details on register notation and possible operand specifications.

LDCB=areaname
Specifies the address of the LDCB (Librarian Data Control Block) for the request.

CHAINID=chainid
Specifies the address of an area where the sublibrary chain identifier (1 to 8 alphameric characters) is stored by the caller.

CHAIN=chain
Specifies the address of an array of 31 entries with

DS CL7
containing the library name

DS CL8
containing the sublibrary name, which will be passed back with return code 0.

The last valid entry will be followed by a 4-character end indicator: 'FFFFFFFF'X (decimal value: -1). Therefore the size of the storage space given with CHAIN is assumed to be ((7+8)*31 + 4) bytes. If the chain does not exist, the end indication will be moved to the start of the area.

The CHAIN operand is required if SCOPE=TASK has been specified or if the CATALOG operand is missing.

CATALOG=sublib

Specifies the address of a field that is used to retrieve the name of the CATALOG sublibrary from the external Job Control LIBDEF CATALOG statement (only if SCOPE=JOB or SCOPE=PART). If SCOPE=TASK, the operand is ignored.

The format of the field is: DS CL7 taking up the library name DS CL8 taking up the sublibrary name.

If the CATALOG sublibrary does not exist, a 4-character 'FFFFFFFF'X (decimal value: -1) end indicator will be moved to the start of the area.

SCOPE=TASK | JOB | PART

Defines the scope associated with the chain:

PART
Indicates the chains which will be kept until partition deactivation or re-definition (LIBDEF...PERM).

JOB Indicates the chains which will be kept until job termination or re-definition.

TASK
Indicates the chains which will be kept until task termination or re-definition (LIBRM LIBDEF).

The default is SCOPE=TASK.

NOTFND=label

Specifies a label to which the service will branch if the specified chain does not exist (return code 8).

EROPT=RET | CANCEL

Defines an error handling option which will be taken if the function cannot be performed (return code > 12).

RET
Processing will be continued, either by a normal return or by branching to the ERRAD exit.

CANCEL
Processing will be canceled. A librarian error message will be issued to SYSLOG.

ERRAD=label

Specifies a label to which the Librarian will branch if the above function cannot be performed because of an error (return code > 12).

Return Codes

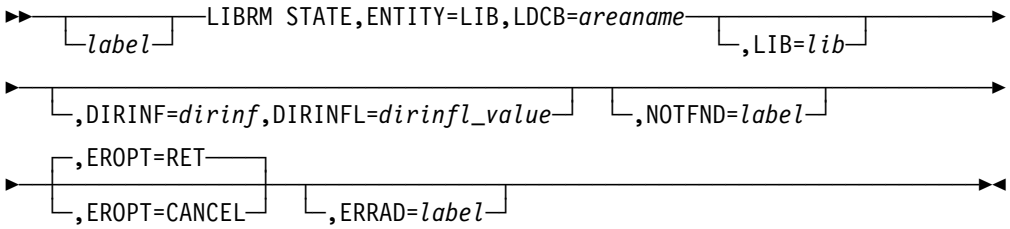
Return Code	Reason Code	Meaning
0	0	Chain exists, information is returned in CHAIN.
8	0	The chain does not exist.
16	xx	External system error with feedback code and message.
20	xx	Internal system error with feedback code and message.
32	0	Access control failed (with message L163I).

The external system error feedback codes are described in Appendix D, “Librarian Feedback Codes” on page 455. All other (internal) feedback codes are described in *VSE Central Functions Librarian Diagnosis Reference*, SC33-6330.

Notes:

1. A generic specification for CHAINID is not possible.
2. If both the CATALOG and the CHAIN operand are specified, the NOTFND condition (return code 8) will only be true if *both* requested chains are absent.
3. The STATE function will always return complete sublibrary names with one exception: For requests with CHAINID 'PHASE' and SCOPE=JOB, which reflects the JCL LIBDEF PHASE,SEARCH....,TEMP statement, it is possible that a special entry for the system directory list (SDL) is returned. The SDL is not a real sublibrary but a reference to the 'virtual' phase library in the SVA. It cannot be processed by any other library services. In this case the STATE function returns the string 'SDL' as a library name and a string with blanks (40'X) as sublibrary name.

LIBRM STATE LIB (Search Library) Macro



Requirements for the caller:

AMODE: 24

RMODE: 24

ASC Mode: Primary

The STATE (LIB) function is used to determine whether a particular library exists. If it exists, the library attributes will be provided in the area specified by DIRINF and DIRINFL (DSECT INLCLBST).

The status information for a library is variable in size. It starts with logical library attributes like 'number of sublibs' or 'number of used (free) blocks' as a fixed part. One to sixteen extent information records may then follow. The extent information describes the physical location of the library. The number of returned extent records is provided in the halfword LBSTXTRT.

The LIB operand cannot be specified generically.

See also “Library Macro Notation” on page 253 for details on register notation and possible operand specifications.

LDCB=areaname
Specifies the address of the LDCB (Librarian Data Control Block) for the request.

LIB=lib
Specifies the address of an area where the library name (1 to 7 alphameric characters) is stored. A generic specification for the library name is not possible.

DIRINF=dirinf
Specifies the address of an area where library descriptor information is returned.

DIRINFL=dirinfl-value
Specifies the length (numeric value) of the DIRINF area.

NOTFND=label
Specifies a label to which the service will branch if the library does not exist or is not accessible (return code 8).

EROPT=RET | CANCEL
Defines an error handling option which will be taken if the function cannot be performed (return code > 12).

RET
Processing will be continued, either by a normal return or by branching to the ERRAD exit.

CANCEL
Processing will be canceled. A librarian error message will be issued to SYSLOG.

ERRAD=label
Specifies a label to which the Librarian will branch if the above function cannot be performed because of an error (return code > 12).

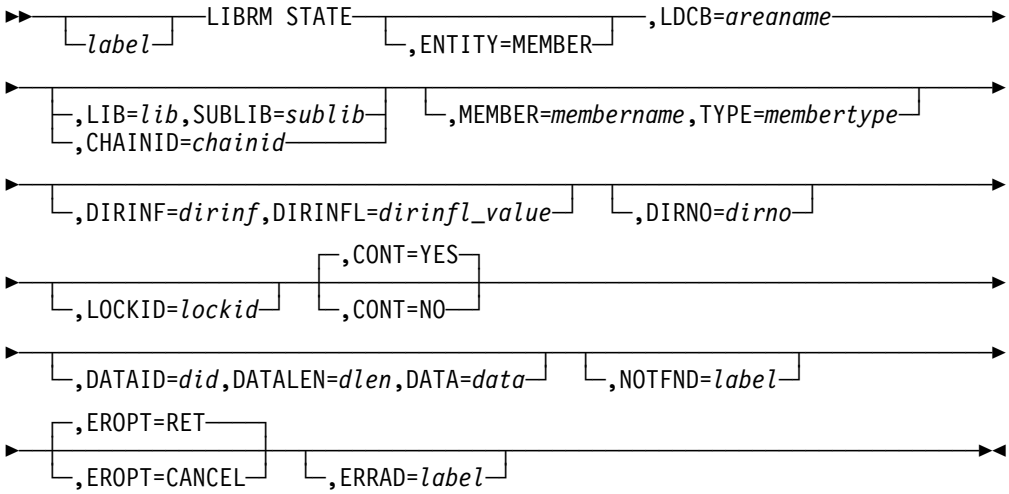
Return Codes

Return Code	Reason Code	Meaning
0	0	Library exists, information is returned in DIRINF.
	4	Library exists, but no information is returned, because DIRINF was not specified.
4	0	Library exists, but the area is too small to return all available extent records. LBSTXTRT contains the number of records actually returned. LBSTXTRT can be zero.
	4	Library exists, but the area is too small to hold even the fixed part of the status information. No information is returned at all.
8	0	The specified library does not exist.
16	xx	External system error with feedback code and message.
20	xx	Internal system error with feedback code and message.
32	0	Access control failed (with message L163I).

The external system error feedback codes are described in Appendix D, “Librarian Feedback Codes” on page 455. All other (internal) feedback codes are described in *VSE Central Functions Librarian Diagnosis Reference*, SC33-6330.

Note: To have access to a library, the library must exist on a volume accessible by the system, and the user must provide the correct labels for it. Otherwise, the library is treated as nonexistent.

LIBRM STATE MEMBER (Search Library Member) Macro



Requirements for the caller:

AMODE: 24

RMODE: 24

ASC Mode: Primary

The STATE (MEMBER) function is used to determine whether a particular member exists in a certain sublibrary or a chain of sublibraries. If it is found, the member attributes will be provided in the area specified by DIRINF and DIRINFL (DSECT INLCMBST).

The MEMBER and TYPE operands can be specified generically by giving a prefix of the name followed by an asterisk (for example, PREF*). In this case, all members whose names start with PREF will be returned. If only * (without a prefix) is specified, **all** members will be returned. For generic requests with chain processing, the sublibrary with the first match will be selected and all matching members within this sublibrary will be returned.

The LOCKID operand specifies the lock identifier of the library member to be searched for. The state of the specified member will be displayed only if the member is locked with this lockid. Otherwise, return code 8 is issued.

The status information entries of the members are stored contiguously in the area specified by DIRINF and DIRINFL. The number of returned entries can be retrieved via DIRNO. If the given space is insufficient, return code 4 is issued. A continuation request is possible.

The library access service will treat the logically next call as a continuation request under the following conditions:

1. The previous request requires a continuation (return code 4, reason code 0).
2. The option CONT=YES is set.
3. The LDCB still defines a STATE,ENTITY=MEMBER function.

The caller should not change the LDCB (and any fields referenced by the LDCB) while processing continuation requests.

The continuation logic will be reset if (1) the space defined by DIRINF and DIRINFL is sufficient to hold the returned members (return code 0), or (2) a processing error occurs (return code > 12), or (3) the caller switches to CONT=NO.

User-provided directory information identified by DATAID is returned in DATA, its length in DATALEN. This service is only provided for the STATE of a single member (nongeneric).

See also “Library Macro Notation” on page 253 for details on register notation and possible operand specifications.

LDCB=areaname
Specifies the address of the LDCB (Librarian Data Control Block) for the request.

LIB=lib
Specifies the address of an area where the library name (1 to 7 alphameric characters) is stored.

SUBLIB=sublib
Specifies the address of an area where the sublibrary name (1 to 8 alphameric characters) is stored.

CHAINID=chainid
Specifies the address of an area where the sublibrary chain identifier (1 to 8 alphameric characters) is stored.

MEMBER=membername
Specifies the address of an area where the member name (1 to 8 alphameric characters) is stored.

TYPE=membertype
Specifies the address of an area where the member type (1 to 8 alphameric characters) is stored.

DIRINF=dirinf
Specifies the address of the area where member directory information is returned.

DIRINFL=dirinfl-value
Specifies the length (numeric value) of the DIRINF area.

DIRNO=dirno
Specifies the address of a fullword where the number of returned directory entries is provided.

LOCKID=lockid
Specifies the lock identifier of the library member to be searched for. The state of the specified member will be displayed only if the member is locked with this **lockid**. Otherwise, return code 8 is given indicating that no member locked with the specified lockid was found. If the member is locked, the **lockid** is copied into the area specified by DIRINF and DIRINFL. The **lockid** (provided in the LIBRM LOCK macro) is a string of up to eight alphanumeric characters.

The **lockid** can also be specified generically by giving a prefix, followed by an asterisk (for example, PRE*). In this case, information for the specified member will be returned only if it is locked with a **lockid** starting with the specified prefix. If LOCKID=* is specified, the member information will be returned only if the member is locked at all (with any **lockid**).

The **lockid** in the LDCB will be cleared after processing.

CONT=YES | NO
Defines whether the continuation should be requested (CONT=YES) or not (CONT=NO). This option will be ignored for a nongeneric specification. CONT=YES is the default.

DATAID=did
Specifies the address of an area where the 1 to 4 character alphameric identifier for user-provided directory information is specified.

DATALEN=dlen
Specifies the address of a fullword where the length of the DATA area is given. The actual length of the user-provided information will be returned by the service.

DATA=data
Specifies the address of the area where the user-provided directory information is returned. The area specified must have a minimal length of DATALEN.

NOTFND=label
Specifies a label to which the service will branch if the member does not exist within the searched sublibraries (return code 8), or the related sublibrary, library or chain does not exit (return code 12).

EROPT=RET | CANCEL
Defines an error handling option which will be taken if the function cannot be performed (return code > 12).

RET
Processing will be continued, either by a normal return or by branching to the ERRAD exit.

CANCEL
Processing will be canceled. A librarian error message will be issued to SYSLOG.

ERRAD=label
Specifies a label to which the Librarian will branch if the above function cannot be performed because of an error (return code > 12).

Return Codes

Return Code	Reason Code	Meaning
0	0	Member exists (or generic members exist), information is returned in DIRINF.
	4	Member exists (or generic members exist), but no information is returned because DIRINF was not specified.
4	0	Match found for generic specification, but DIRINF is too small to contain all entries. Continuation is possible.
	4	Match found for nongeneric or generic specification, but area is too small for a full entry. No information is returned. Continuation is not possible.
8	0	No match found.
12	0	The specified sublibrary does not exist.
	4	The specified library does not exist.
	8	The specified chain does not exist.
16	xx	External system error with feedback code and message.
20	xx	Internal system error with feedback code and message.
32	0	Access control failed (with message L163I).

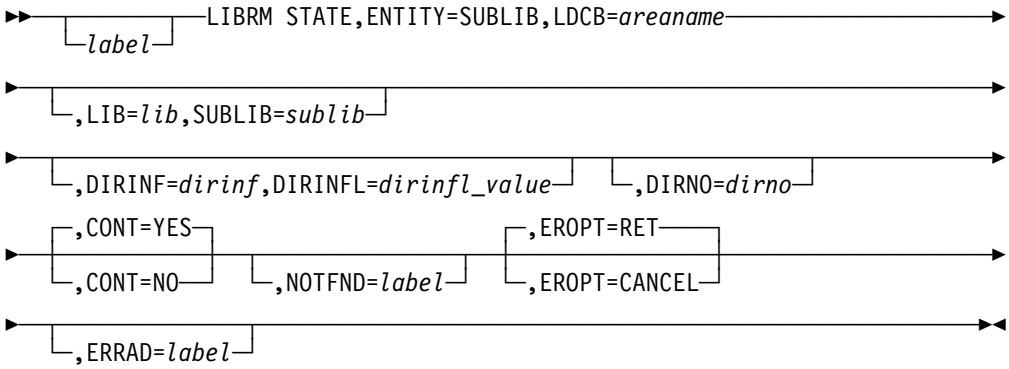
The external system error feedback codes are described in Appendix D, “Librarian Feedback Codes” on page 455. All other (internal) feedback codes are described in *VSE Central Functions Librarian Diagnosis Reference*, SC33-6330.

Notes:

1. You can read the member directory (all members) sequentially by specifying MEMBER=* and TYPE=*, providing a DIRINF area for one entry, and by continuing (return code 4) until return code 0 is passed with the last entry.
2. If both LIB/SUBLIB and CHAINID are specified, LIB/SUBLIB will be taken and CHAINID will be ignored.
3. If user-provided directory information (operands DATA, DATALEN, DATAID) cannot be returned, the fullword addressed with DATALEN will be set to zero. Return and reason codes are not affected.

   No processing is done for a generic member request and for a nongeneric request if the directory information cannot be returned (return code 0 or 4, with reason code 4).

LIBRM STATE SUBLIB (Search Sublibrary) Macro



Requirements for the caller:

AMODE: 24

RMODE: 24

ASC Mode: Primary

The STATE (SUBLIB) function is used to determine whether a particular sublibrary exists in a certain library. If it is found, the sublibrary attributes will be provided in the area specified by DIRINF and DIRINFL (DSECT INLCSBST).

The SUBLIB operand can be specified generically by giving a prefix of the name followed by an asterisk (for example, PREF*). In this case, all sublibraries whose names start with PREF will be returned. If only * (without a prefix) is specified, **all** sublibraries will be returned.

The status information entries of the sublibraries are stored contiguously in the area specified by DIRINF and DIRINFL. The number of returned entries can be retrieved via DIRNO. If the given space is insufficient, return code 4 is returned. A continuation request is possible.

The library access service will treat the logically next call as a continuation request under the following conditions:

1. The previous call requires a continuation (return code 4, reason code 0).
2. The option CONT=YES is set.
3. The LDCB still defines a STATE,ENTITY=SUBLIB function.

The caller should not change the LDCB (and any fields referenced by the LDCB) while processing continuation requests.

The continuation logic will be reset if (1) the space defined by DIRINF and DIRINFL is sufficient to hold the returned sublibraries (return code 0), or (2) a processing error occurs (return code > 12), or (3) the caller switches to CONT=NO.

See also “Library Macro Notation” on page 253 for details on register notation and possible operand specifications.

LDCB=areaname
Specifies the address of the LDCB (Librarian Data Control Block) for the request.

LIB=lib
Specifies the address of an area where the library name (1 to 7 alphameric characters) is stored.

SUBLIB=sublib
Specifies the address of an area where the sublibrary name (1 to 8 alphameric characters) is stored.

DIRINF=dirinf
Specifies the address of an area where sublibrary directory (INLCSBST) information is returned.

DIRINFL=dirinfl-value
Specifies the length (numeric value) of the DIRINF area.

DIRNO=dirno
Specifies the address of a fullword where the number of directory entries is returned.

CONT=YES | NO
Defines whether continuation should be requested (CONT=YES) or not (CONT=NO). This option will be ignored for an nongeneric specification. CONT=YES is the default.

NOTFND=label
Specifies a label to which the service will branch if the sublibrary does not exist within the given library (return code 8), or the related library does not exit or is not accessible (return code 12).

EROPT=RET | CANCEL
Defines an error handling option which will be taken if the function cannot be performed (return code > 12).

RET
Processing will be continued, either by a normal return or by branching to the ERRAD exit.

CANCEL
Processing will be canceled. A librarian error message will be issued to SYSLOG.

ERRAD=label
Specifies a label to which the Librarian will branch if the above function cannot be performed because of an error (return code > 12).

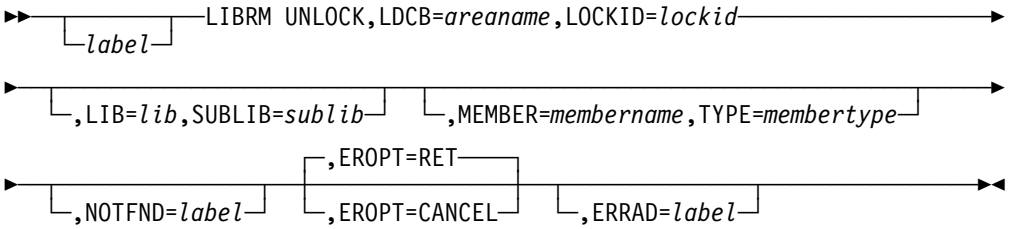
Return Codes

Return Code	Reason Code	Meaning
0	0	Sublibrary exists (or generic sublibraries exist), information is returned in DIRINF.
	4	Sublibrary exists (or generic sublibraries exist), but no information is returned, because DIRINF was not specified.
4	0	Match found for generic specification, but DIRINF is too small to contain all entries. Continuation request is possible.
	4	Match found for nongeneric or generic specification, but the area is too small to contain a full entry. No information is returned. Continuation is not possible.
8	0	No match found.
12	4	The specified library does not exist.
16	xx	External system error with feedback code and message.
20	xx	Internal system error with feedback code and message.
32	0	Access control failed (with message L163I).

The external system error feedback codes are described in Appendix D, “Librarian Feedback Codes” on page 455. All other (internal) feedback codes are described in *VSE Central Functions Librarian Diagnosis Reference*, SC33-6330.

Note: You can read the sublibrary directory sequentially by specifying SUBLIB=*, providing a DIRINF area for one entry, and by continuing (return code 4) until return code 0 is passed with the last entry.

LIBRM UNLOCK (Unlock Library Member) Macro



Requirements for the caller:

AMODE: 24

RMODE: 24

ASC Mode: Primary

The UNLOCK (MEMBER) function causes a library member that has been locked for write or update access to be unlocked again. The member must be fully qualified. It will be unlocked only if the specified **lockid** matches the **lockid** with which this member was locked (with LIBRM LOCK). Only the **lockid** specification may be generic.

See also “Library Macro Notation” on page 253 for details on register notation and possible operand specifications.

LDCB=areaname
Specifies the address of the LDCB (Librarian Data Control Block) for the request.

LOCKID=lockid
Specifies the lock identifier with which the member was locked (with the corresponding LOCK function). The member will be unlocked only if the specified **lockid** corresponds with the **lockid** with which this member was locked. The **lockid** is a string of up to eight alphanumeric characters.

The **lockid** can also be specified generically by giving a prefix, followed by an asterisk (for example, PRE*). In this case, the specified member will be unlocked only if it is locked with a **lockid** starting with the specified prefix. If LOCKID=* is specified, the member will be unlocked if it is locked at all (with any **lockid**).

The **lockid** in the LDCB will be cleared after processing.

LIB=lib
Specifies the address of an area where the library name (1 to 7 alphameric characters) is stored.

SUBLIB=sublib
Specifies the address of an area where the sublibrary name (1 to 8 alphameric characters) is stored.

MEMBER=membername
Specifies the address of an area where the member name (1 to 8 alphameric characters) is stored. No generic specification is allowed.

TYPE=membertype
Specifies the address of an area where the member type (1 to 8 alphameric characters) is stored.

NOTFND=label
Specifies a label to which the service will branch if the specified member does not exist (return code 8 for members opened for INPUT), or if the specified library or sublibrary does not exist or is not accessible (return code 12).

EROPT=RET | CANCEL
Defines an error handling option which will be taken if the function cannot be performed (return code > 12).

RET
Processing will be continued, either by a normal return or by branching to the ERRAD exit.

CANCEL
Processing will be canceled. A librarian error message will be issued to SYSLOG.

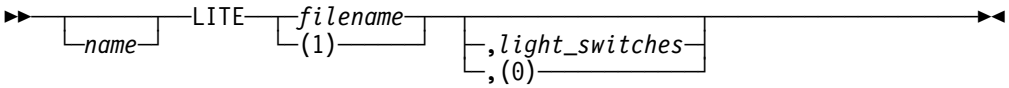
ERRAD=label
Specifies a label to which the Librarian will branch if the above function cannot be performed because of an error (return code > 12).

Return Codes

Return Code	Reason Code	Meaning
0	0	Member has been unlocked.
0	8	The macro is ignored, no member will be unlocked. Option IGNLOCK is on.
4	0	The specified member was already unlocked.
8	0	The specified member does not exist.
	4	The specified member is locked with a different lockid.
12	0	The specified sublibrary does not exist.
	4	The specified library does not exist.
16	xx	External system error with feedback code and message.
20	xx	Internal system error with feedback code and message.
32	0	Access control failed (with message L163I).

The external system error feedback codes are described in Appendix D, “Librarian Feedback Codes” on page 455. All other (internal) feedback codes are described in *VSE Central Functions Librarian Diagnosis Reference*, SC33-6330.

LITE (Pocket-Light Control) Macro



Requirements for the caller:

AMODE: 24

RMODE: 24

ASC Mode: Primary

This macro lights any combination of pocket lights on an IBM 1419 Magnetic Character Reader or an IBM 1275 Optical Reader/Sorter. Before using the LITE macro, the DISEN macro must be issued to disengage the device. Processing of the documents should be continued until the unit exception bit (byte 0, bit 3) of the document buffer status indicators is set on (see Figure 3 on page 70). When this bit is on, the follow-up documents have been processed, the MICR reader has been disengaged, and the pocket LITE macro can be issued.

filename | (1)
Is the name of the file; this name is the same as that specified for the DTFMR header entry for the file.

light-switches | (0)
Indicates a 2-byte area containing the pocket light switches. Both operands can be given either as a symbol or in register notation.

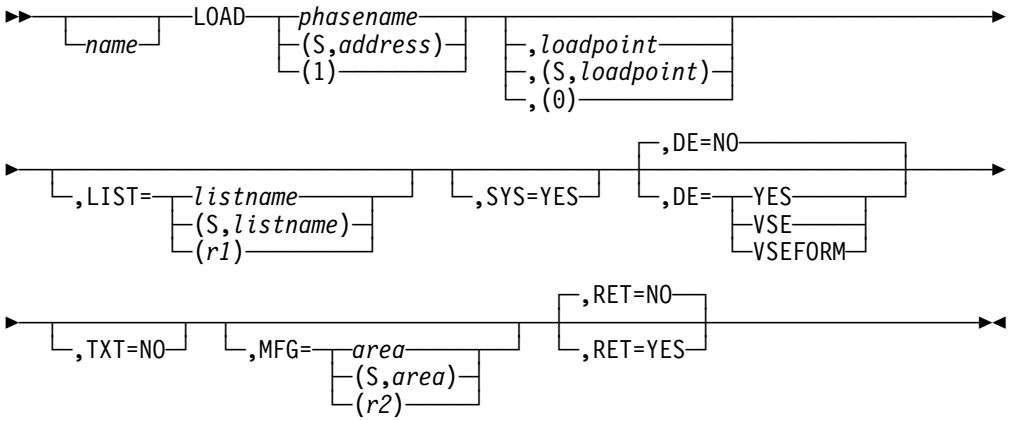
The bit configuration for the pocket light switch area is shown below. The pocket lights that are turned on should have their indicator bits set to 1. If an error occurs during the execution of a pocket lighting I/O command, bit 7 in byte 1 is set to 1. This error condition normally indicates that the pocket light operation was unsuccessful.

The bit configuration of the pocket light switch area is shown in Figure 29.

Byte	Bit	Corresponding Pocket Light	Other Purpose
0	0	A	
	1	B	
	2	0	
	3	1	
	4	2	
	5	3	
	6	4	
	7	5	
1	0	6	
	1	7	
	2	8	
	3	9	
	4-6		Reserved
	7		Error indicator

Figure 29. Bit Configuration of the Pocket-Light Switch Area

LOAD (Load a Phase) Macro



Requirements for the caller:

AMODE: 24 or 31

RMODE: 24

ASC Mode: Primary

The LOAD macro loads the phase specified in the first operand (if this phase is not in the SVA) and returns control to the calling program.

After execution of the macro, the entry-point address of the called phase is returned to you in register 1. LOAD sets the high-order bit in register 1 to indicate the phase's AMODE (0 for 24, 1 for 31). If the phase's AMODE is ANY, LOAD sets the high-order bit in register 1 corresponding to the caller's AMODE.

Notes:

1. Both the expanded code and the parameter lists must be below 16MB, otherwise the request will be canceled.
2. The addresses are not validated by the macro expansion. Depending on the macro call, only three bytes of a passed four-byte address may be passed to the SVC service. This may cause unpredictable results, since the address cannot be validated properly.

For a nonrelocatable phase, the address of the called phase is the entry-point determined at link-edit time. For a relocatable phase, the entry point is adjusted by the relocation factor.

If the phase is in the SVA, it is not loaded. Instead, the system returns, in register 1, the entry-point address of the phase in the SVA. If, however, the SDL operand is specified in the LIBDEF statement, the phase is loaded if it is found in one of the sublibraries specified before the SDL operand.

phasename | (S,address) | (1)

For phasename specify the name of the required phase. The address is regarded as either a 24-bit or 31-bit address, depending on the AMODE of the caller. If the caller has AMODE 31 and the address points to a storage location above 16MB, the requestor is canceled.

If the DE operand is omitted or if DE=NO is specified, the address as specified in (S, address) or as loaded into a register points to an 8-byte field that contains the phase name.

If DE=YES or DE=VSE | VSEFORM, the operand has a different meaning; refer to the discussion of the DE operand.

loadpoint | (S,loadpoint) | (0)
If loadpoint is provided, the phase is loaded at the specified address. If no loadpoint is given, the (relocated) loadpoint specified at link-edit time is used. The loadpoint is interpreted as either a 24-bit or 31-bit address, depending on the AMODE of the caller.

The address used must be outside the supervisor area. When an overriding address is given, the entry-point address is relocated and returned in register 1. An overriding load-point address must not be specified for a phase that had been linked as a member of an overlay structure.

If the phase is non-relocatable, none of the other addresses in the phase are relocated; if the phase is relocatable, however, the entry point and address constants are updated with the relocation factor.

If loadpoint is given in register notation, the register used must not be register 1. Pre-load the register with the load-point address.

With (S,...) notation, the load-point address is derived from base register and displacement as assembled for loadpoint in the (S,loadpoint) specification.

LIST=listname | (S,listname) | (r1)
For listname specify the name of your local directory list generated in the partition by the GENL macro. When this operand is included, the system scans the local directory list for the name of the required phase before it initiates a search for this phase name in the directories of accessible sublibraries.

If the phase has been found in the local directory list, general register 0 points to the related directory entry; otherwise, register 0 is set to zero.

The local directory list must be located below 16MB (only three bytes are used in the macro expansion).

If LIST is specified, DE=YES or DE=VSE | VSEFORM is invalid.

SYS=YES
If SYS=YES is specified, the system scans the system directory list (SDL) in the SVA and the system sublibrary before any private sublibraries. If the operand is omitted, the SDL and the private sublibraries are searched first.

DE=NO | YES | VSE | VSEFORM
By specifying DE=YES or DE=VSE | VSEFORM you can generate your own local directory entry for a frequently used phase in order to save a time-consuming library directory search for that phase. A specification of YES or VSE|VSEFORM is invalid if LIST is specified.

DE=NO Indicates that no local directory entry is to be generated.

DE=YES Indicates a conventional 38-byte directory entry in the old (VSE/Advanced Functions Version 1) librarian format is to be generated.

DE=VSE | VSEFORM
Indicates that a 40-byte directory entry in the new (VSE/Advanced Functions Version 2) librarian format is to be generated. VSE is a short form of VSEFORM.

For DE=YES or DE=VSE, the MAPDNTRY macro can be used to interpret the information returned by the LOAD (or FETCH) macro. Among other information, the local directory entry shows the AMODE/RMODE assigned to the phase. The directory entry must be located below 16MB (see explanation for **phasename**).

The local directory entry is activated by the first LOAD request; all further LOAD requests are executed without any directory search.

If the first operand is written as phasename (instead of S-type or register notation), a directory entry will be generated within the macro expansion. The generated directory entry will contain the name of the phase in the first eight bytes.

If you use S-type or register notation for the first operand, you must set aside the 38-byte (or 40-byte) field for the directory entry yourself and point to it via this operand. The directory entry must contain the phase name in the first 8 bytes (left-justified and padded with blanks); its format is:

For DE=YES:

```
Bytes    Contents
  0      CL8'PHASENAM'
  8      XL3'0'
 11      XL1'0D'      NO. OF HALFWORDS FOLLOWING
 12      XL26'0'
```

For DE=VSE | VSEFORM:

```
Bytes    Contents
  0      CL8'PHASENAM'
  8      XL3'FFFFFF'      ID FOR NEW FORMAT
 11      XL1'0E'          NO. OF HALFWORDS FOLLOWING
 12      XL28'0'
```

TXT=NO
Together with LIST=listname or DE=YES, TXT=NO is useful if a phase is to be loaded more than once while your program executes. TXT=NO causes a search for the directory entry without transfer of the contents (or text) of the phase itself. It indicates, in the directory entry, if and where the phase was found. This can be used to accomplish either of the following:

- The directory entry can be filled in from the sublibrary for later FETCH/LOAD calls without the overhead of text transfer.
- You can establish whether a given phase is present in a user sublibrary, or the SYSLIB sublibrary, or the SVA since register 0 contains the address of the directory entry and byte 16 of the directory entry is:

  X'06' If the phase is not found
  X'12' If the phase is in the SVA
  X'0A' If the phase is in a user sublibrary

Note: Test for these conditions by means of a Test Under Mask (TM) instruction, not a Compare instruction. If the phase is not found and

both DE=YES and TXT=NO have been specified, register 1 is returned with X'00'. See Figure 34 on page 304.

MFG=area | (S,area) | (r2)

The operand is required if the program that issues the LOAD macro is to be reenterable. It specifies the address of a 64-byte dynamic storage area, that is, storage which your program obtained through a GETVIS macro. This area is required for system use during execution of the macro.

The MFG area must be located below 16MB.

RET=NO | YES

By specifying RET=YES you can cause control to be returned to your program in any case (both in normal and error situations). Register 15 contains one of the following return codes:

0 LOAD completed successfully.

4 Phase not found. This return code is issued also if a user directory entry is found and the corresponding phase has already been deleted or re-cataloged.

8 Irrecoverable I/O error during LOAD processing.

12 Invalid library or sublibrary structure detected during LOAD processing.

16 Either of the following was found during LOAD processing:

- Local directory entry outside the partition.
- Phase does not fit into the partition.
- Loadpoint outside partition.

20 Security violation.

24 Inconsistent user directory state − LOAD found an inconsistency between your program's local directory entry and the corresponding entry in the directory of the related sublibrary. The local directory entry is overwritten by the entry read from the directory of the sublibrary. LOAD checks the following:

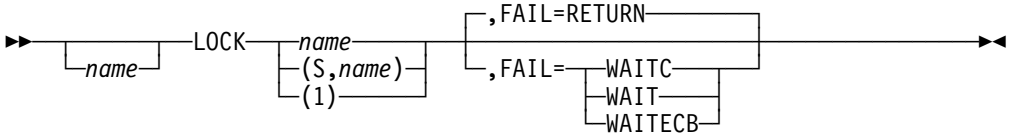
- Length of phase.
- Relocation state.
- Difference between the load point and the partition start address.
- Difference between the load point and the entry point.

Return code 24 is also issued if an incorrect length has been specified in byte 11 of the directory entry.

28 Partition is too small (or phase does not fit into the logical transient area).

36 A loadpoint is provided that causes a mismatch with the RMODE specification in the phase's local directory (that is, loadpoint and/or end of phase is above 16MB and RMODE=24). The phase is not loaded. If RET=NO is specified, the user is canceled with 'RMODE violation'.

LOCK (Lock a Resource) Macro



Requirements for the caller:

AMODE: 24 or 31

RMODE: 24

ASC Mode: Primary

The macro queues the task for accessing the named resource. The resource must have been defined in the program by a DTL (Define The Lock) control block. A DTL is generated by issuing a DTL or GENDTL macro; it may be modified by issuing a MODDTL macro.

Note: Do not LOCK a resource in an AB exit routine if this resource is held by the main task, since a deadlock situation may occur.

name | (S,name) | (r)
Specifies the DTL address.

FAIL=RETURN | WAITC | WAIT | WAITECB
Defines the system action in case the resource cannot be obtained:

FAIL=RETURN
Causes the system to return control back to the requesting program in any case. The requesting program has to check the return code in register 15 to find out whether or not the request was successful.

FAIL=WAITC
Causes the system to place the requesting task in the wait state if the requested resource is found to be locked by another task. In all other cases, control returns to the requesting program. The requesting program has to check the return code in register 15 to find out whether the request was successful, or whether an error occurred.

FAIL=WAIT
Requests the system to return control to the requesting task when the resource can be obtained. If the resource is locked by another task, the requesting task is set into the wait state until the resource is freed. In case of an error condition with a return code of 12 or higher, the requesting task is canceled.

FAIL=WAITECB
Causes the system to place the requesting task in the request queue if the requested resource is found to be locked by another task. In all cases, control returns to the requesting program. The requesting program has to check the return code in register 15 to find out whether the request was successful, or whether an error occurred, or whether the task's request is put in the request queue (return code = 4). If the

request was put in the request queue, the requesting task should issue a WAIT macro on this ECB.

WAIT, WAITC or WAITECB cannot be specified if the resource is defined with OWNER=PARTITION.

Figure 30 summarizes how the system controls access to a resource, depending on the specification of the CONTROL and LOCKOPT operands in the DTL or GENDTL macro. The illustration assumes that a task issues a LOCK request for a resource which is already locked.

	Current Lock Status of Resource					
	LOCKOPT=1 CONTROL=		LOCKOPT=2 CONTROL=		LOCKOPT=4 CONTROL=	
Incoming LOCK Request	E	S	E	S	E	S
LOCKOPT=1 CONTROL=E	W	W	W	W	W	W
CONTROL=S	W	G	I	I	I	I
LOCKOPT=2 CONTROL=E	W	I	W	G	I	I
CONTROL=S	W	I	G	G	I	I
LOCKOPT=4 CONTROL=E	W	I	I	I	G/W	G
CONTROL=S	W	I	I	I	G	G

where:

G = Access is granted (return code 0).

I = Access is not granted. The incoming LOCK request is inconsistent with the current LOCK status (return code 12).

W = Access to the resource cannot be granted (return code 4 or 16).

G/W = Access is granted if the resource is already exclusively owned by the requesting system. Access is denied (return code 4) if the resource is exclusively held by the other system.

Figure 30. System Action for Control Definitions in DTLs

A task or partition may lock a resource more than once. The system maintains a lock request count for the resource.

When a resource is defined with LOCKOPT=1, a task may issue up to 255 LOCK requests with CONTROL=S. When a resource is defined with LOCKOPT=2, up to 255 LOCK requests with CONTROL=S and (if no other task locks the resource exclusively) one LOCK request with CONTROL=E are allowed.

When a resource is locked more than once by a task, this task has to issue at least as many UNLOCK requests as it issued LOCK requests before it gives up the resource completely. If the resource is defined with OWNER=PARTITION, the unlocking may be done by any task in the partition.

Return Codes in Register 15

Figure 31 gives a summary of system actions by return codes, depending on the specification of the FAIL operand. A list of possible return codes together with a summary of their meanings follows Figure 31. If FAIL=WAITECB is specified, the return code is also presented in byte 1 of the ECB.

Return Code		System Action if			
Hex	Dec.	FAIL=RETURN	FAIL=WAITC	FAIL=WAIT	FAIL=WAITECB
00	0	Return	Return	Return	Return
04	4	Return	Wait	Wait	Queue + Return
08	8	Return	Return	Wait	Return
0C	12	Return	Return	Cancel	Return
10	16	Return	Return	Cancel	Return
14	20	Return	Return	Cancel	Return
18	24	Return	Return	Cancel	Return
1C	28	Return	Return	Wait	Return
20	32	Return	Return	Cancel	Return
24	36	Return	Return	Cancel	Return

Figure 31. System Actions by Return Code and FAIL Operand

0 Successful request: the resource is locked for the task (or for the partition if the resource is defined with partition ownership).

4 Resource not available: the resource is already locked with a locking status that allows no concurrent access.

8 The lock table space is exhausted.

12 The lock request is inconsistent with previous lock requests (by the same or other tasks).

16 The request would have resulted in a deadlock condition within the system (deadlocks across systems are not affected).

20 DTL format error.

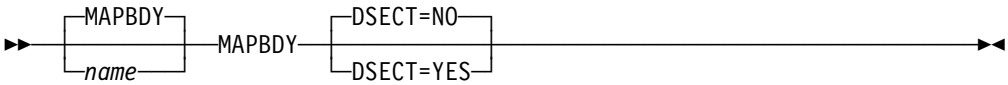
24 The issuing task tried to lock a resource which it owns already exclusively. Note that the locking status may not be the same as the one specified in the DTL macro with the CONTROL and LOCKOPT options.

28 The lock request resulted in a lock file overflow condition. Use the DLF command to specify a larger size for the lock file.

32 A lock request was issued for a shared DASD file, but the corresponding volume is not online.

36 An unrecoverable I/O error occurred on the lock file. This probably means that the system must be restarted and the lock file re-defined. This has to be done on all sharing systems.

MAPBDY (Map Boundary Information) Macro



Required RMODE: **24** or **ANY**

The macro may be used to interpret the information retrieved by the EXTRACT macro for ID=BDY and MODE=T. If 'name' is omitted, MAPBDY is taken as default.

DSECT=NO | YES

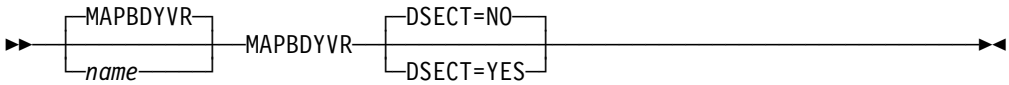
DSECT=YES specifies that a mapping DSECT is generated. If the operand is omitted, inline code is generated.

For the layout and a description of the DSECT fields, see Figure 32.

Field Name	No. of Bytes	Contents
PBEGIN	4	Partition start address; corresponds to the address of the program save area (field SAVE of PIB).
PENDLOG	4	Logical end of partition (last addressable byte, GETVIS area excluded); corresponds to PPEND of the partition communication region.
PGEND	4	Physical end of partition (last addressable byte, GETVIS area included).
PFIXLMT	4	PFIX limit (in No. of K bytes) or zero (in real mode).
PFIXCNT	4	PFIX count (No. of pages fixed by a PFIX request).

Figure 32. Layout of the MAPBDY-Generated DSECT

MAPBDYVR (Map Boundary Information) Macro



Required RMODE: **24** or **ANY**

The MAPBDYVR macro may be used to interpret the information retrieved by the EXTRACT macro for ID=BDY and MODE=P. If 'name' is omitted, MAPBDYVR is taken as default.

DSECT=NO | YES

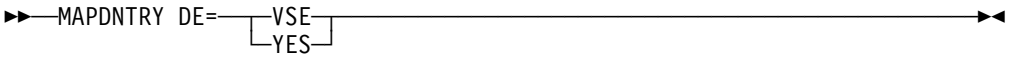
DSECT=YES specifies that a mapping DSECT is generated. If the operand is omitted, inline code is generated.

For the layout and a description of the DSECT fields, see Figure 33.

Field Name	No. of Bytes	Contents
VPBEGIN	4	Virtual partition start address; corresponds to the latest allocation.
VPEND	4	Logical end of virtual partition (last addressable byte, GETVIS area excluded).
VPGEND	4	Physical end of virtual partition (last addressable byte, GETVIS area included); corresponds to the latest allocation.
RPBEGIN	4	Real partition start address; corresponds to the latest allocation.
RPEND	4	Real partition end address (last addressable byte); corresponds to the latest allocation.

Figure 33. Layout of the MAPBDYVR-Generated DSECT

MAPDNTRY (Map Directory Entry) Macro



The MAPDNTRY macro generates a mapping DSECT which may be used to interpret the information returned by the LOAD or FETCH macro when DE=VSE or DE=YES has been specified.

DE=VSE | YES

DE=VSE specifies the directory entry that will be returned when DE=VSE was specified in the LOAD or FETCH macro. For the layout and a description of the DSECT fields, see Figure 34.

DE=YES specifies the directory entry that will be returned when DE=YES was specified in the LOAD or FETCH macro. For the layout and a description of the DSECT fields, see Figure 35 on page 305.

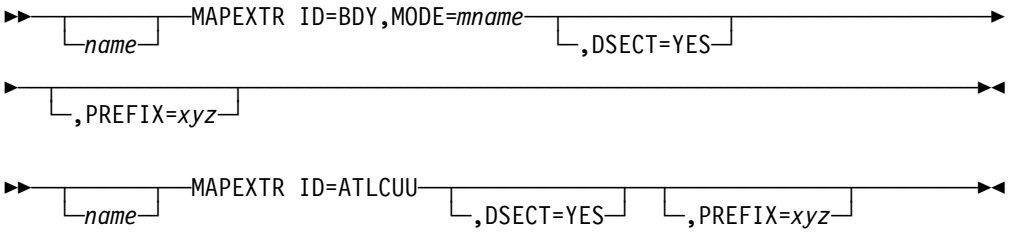
Field Name	No. of Bytes	Contents
DIRNAME	8	Phasename
DIRIDVSE	3	VSE-ID (X'FFFFFF')
DIRN	1	Number of halfwords following
DIRLMBR	4	Length of library member (phase)
DIRC	1	Flags:
SELFREL		X'80' Phase is self-relocatable
RELPHASE		X'40' Phase is relocatable
SVAELIG		X'20' Phase is SVA-eligible
SVAPHASE		X'10' Phase is located in SVA
PCIL		X'08' Non-SYSLIB phase
NOTFND		X'04' Phase not found
ACTIVE		X'02' Directory entry active
DIRSWIT	1	Flags:
DIRRMOD		X'20' 1: RMODE=ANY, 0: RMODE=24
DIRAM31		X'10' 1: AMODE=31 or AMODE=ANY
DIRAM24		X'08' 1: AMODE=24 or AMODE=ANY
		11: AMODE=ANY, 10: AMODE=31
		00 or 01: AMODE=24
	2	Reserved
DIRACOPY	4	Address of directory copy
DIRALPT	4	Load point at link-edit time
DIRAEPT	4	Entry point at link-edit time
DIRAPART	4	Partition start at link-edit time
DIRASVA	4	Entry point of phase in SVA

Figure 34. Layout of the MAPDNTRY-Generated DSECT for DE=VSE

Field Name	No. of Bytes	Contents
DIRNAME	8	Phasename
DIRCOPY	3	Address of directory copy
DIRN	1	Number of halfwords following
DIRTT	2	Number of text blocks
DIRLL	2	Number of text bytes in last block
DIRC	1	Flags:
SELFREL		X'80' Phase is self-relocatable
RELPHASE		X'40' Phase is relocatable
SVAELIG		X'20' Phase is SVA-eligible
SVAPHASE		X'10' Phase is located in SVA
PCIL		X'08' Non-SYSLIB phase
NOTFND		X'04' Phase not found
ACTIVE		X'02' Directory entry active
DIRSWIT	1	Flags:
DIRRMOD		X'20' 1: RMODE=ANY, 0: RMODE=24
DIRAM31		X'10' 1: AMODE=31 or AMODE=ANY
DIRAM24		X'08' 1: AMODE=24 or AMODE=ANY 11: AMODE=ANY, 10: AMODE=31 00 or 01: AMODE=24
DIRPPP	3	Load point at link-edit time
DIREEE	3	Entry point at link-edit time
	3	Reserved
DIRAAA	3	Partition start at link-edit time
DIRVEE	4	Entry point of phase in SVA
	4	Reserved

Figure 35. Layout of the MAPDNTRY-Generated DSECT for DE=YES

MAPEXTR (Map EXTRACT Service) Macro



Required RMODE: **24** or **ANY**

The MAPEXTR macro may be used to interpret the information retrieved by the EXTRACT macro with the corresponding keywords ID=BDY,MODE=mname or ID=ATLCUU. If 'name' is omitted, MAPEXTR is taken as default.

ID=BDY
Specifies that partition boundary information retrieved by the EXTRACT ID=BDY,... macro is to be interpreted.

ID=ATLCUU
Specifies that input required for, and the output returned by, the EXTRACT ID=ATLCUU,... macro is to be mapped. For the layout of the generated DSECT see Figure 39 on page 308.

MODE=mname
Specifies the type of boundary information that has been extracted. **mname** can be one of the following:

T Temporary boundary information retrieved by (old) EXTRACT ID=BDY,MODE=T and mapped by (old) MAPBDY macro. The PREFIX operand is invalid.

TEMP Temporary boundary information retrieved by (new) EXTRACT ID=BDY,MODE=TEMP. For the layout of the generated DSECT see Figure 36 on page 307.

P Permanent boundary information retrieved by (old) EXTRACT ID=BDY,MODE=P and mapped by (old) MAPBDYVR macro. The PREFIX operand is invalid.

PERM Permanent boundary information retrieved by (new) EXTRACT ID=BDY,MODE=PERM. For the layout of the generated DSECT see Figure 37 on page 307.

SYSP System boundary information retrieved by EXTRACT ID=BDY,MODE=SYSP. For the layout of the generated DSECT see Figure 38 on page 308.

DSECT=YES
Specifies that a mapping DSECT is to be generated. If the operand is omitted, inline code will be generated.

PREFIX=xyz
Provides the possibility to change the first three characters of the label 'name'. The prefix value must not be longer than three bytes. If the operand is not specified, it defaults to MAP.

PREFIX is valid for MODE=TEMP|PERM|SYSP only; it is ignored for MODE=T|P, since this invokes the old mapping macros MAPBDY and MAPBDYVR only.

Field Name	No. of Bytes	Contents	
xyzPTBEG	4	Partition start address (first addressable byte)	OLD MAPBDY
xyzPTPPE	4	Logical end of partition (last addressable byte, GETVIS area excluded)	
xyzPTEND	4	Physical end of partition (last addressable byte, GETVIS area included)	
xyzPFXLL	4	PFIX limit (in no. of K bytes) or zero (in real mode)	
xyzPFXCL	4	PFIX count (no. of pages fixed by a PFIX request)	
xyzPFXL3	4	PFIX limit for storage above 16MB	NEW
xyzPFXC3	4	PFIX count for this storage (pages)	
xyzBDYTL		Length of area	

Figure 36. Layout of the MAPEXTR-Generated DSECT for MODE=TEMP. Note that the upper part of the DSECT is identical with the old MAPBDY DSECT.

Field Name	No. of Bytes	Contents	
xyzVPBEG	4	Virtual partition start address; corresponds to the latest allocation.	OLD MAPBDYVR
xyzVPPPE	4	Logical end of virtual partition (last addressable byte, GETVIS area excluded).	
xyzVPEND	4	Physical end of virtual partition (last addressable byte, GETVIS area included); corresponds to the latest allocation.	
xyzRPBEG	4	Real partition start address; corresponds to the latest allocation.	
xyzRPEND	4	Real partition end address (last addressable byte); corresponds to the latest allocation.	
xyzPFXLL	4	PFIX limit for storage below 16MB	NEW
xyzPFXL3	4	PFIX limit for storage above 16MB	
xyzPDYPL		Length of area	

Figure 37. Layout of the MAPEXTR-Generated DSECT for MODE=PERM. Note that the upper part of the DSECT is identical with the old MAPBDYVR DSECT.

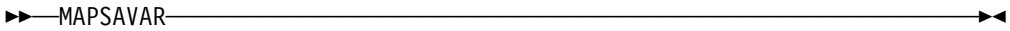
Field Name	No. of Bytes	Contents	
xyzSUPBG	4	Begin of supervisor area	
xyzSUPND	4	End of supervisor area	
xyzSSBEG	4	Begin of SDAID area	O
xyzSSEND	4	End of SDAID area	L
xyzSVA	4	Begin of Shared Virtual Area	D
xyzESVA	4	End of Shared Virtual Area	
xyzSPBEG	4	Begin of shared partition area	M
xyzSPEND	4	End of shared partition area	A
xyzPRPBG	4	Begin of private partition area	P
xyzPRPND	4	End of private partition area	S
xyzVPBEG	4	Begin of VPOOL area	Y
xyzVPEND	4	End of VPOOL area	S
xyzSVIS	4	Begin of system GETVIS area	P
xyzSVISE	4	End of system GETVIS area	
xyzPPBEG	4	Begin of problem program area (may be shared or private)	
xyzSVA3B	4	Begin of 31-bit SVA	
xyzSVA3E	4	End of 31-bit SVA	
xyzSGV3B	4	Begin of 31-bit system GETVIS area	N
xyzSGV3E	4	End of 31-bit system GETVIS area	E
xyzVLB3B	4	Begin of 31-bit virtual library	W
xyzVLB3E	4	End of 31-bit virtual library	
xyzVSIZE	4	Total virtual storage size in KB	I
xyzSYSVS	4	Total virtual storage size of system-used address spaces (KB)	N F
xyzSPFLL	4	System PFIX limit in low area	O
xyzSPFCL	4	System PFIX count in low area	
xyzSPFL3	4	System PFIX limit in high area	
xyzSPFC3	4	System PFIX count in high area	
xyzSYLEN		Length of area	

Figure 38. Layout of the MAPEXTR-Generated DSECT for MODE=SYSP

Field Name	No. of Bytes	Contents
xyzATVER	4	Version indicator, must be X'0000'.
xyzATCUU	2	Cleared to X'0000' and filled with X'0cuu' if cuu was found. (X'0000' if no free drive found, indicated by return code.)
xyzATDVT	6	External device type code of the requested tape drive as on the ADD command (3490E or TPA). Start always at the beginning of the 6 character field (left margin) and continue the input without using blanks (Not used if running under VM, but validity is checked.)
xyzATLNM	8	Logical name of the requested automatic tape library. (Not used if running under VM, but validity is checked.)
xyzATLLN		Length of one entry in list.

Figure 39. Layout of the MAPEXTR-Generated DSECT for ID=ATLCUU

MAPSAVAR (Map Save Area) Macro



The MAPSAVAR macro may be used to interpret the save area information returned by the STXIT macro. A mapping DSECT with the name SVUARA is generated.

For the layout and a description of the DSECT fields, see Figure 40.

Field Name	No. of Bytes	Contents
SVUPSW	4	First half of (BC-mode) PSW of interrupted program
SVUPSW2	4	Second half of PSW
SVUR00	4	Save area for register 0
SVUR01	4	Save area for register 1
.		.
.		.
.		.
SVUR0F	4	Save area for register 15
SVUOLDLN		Old save area length

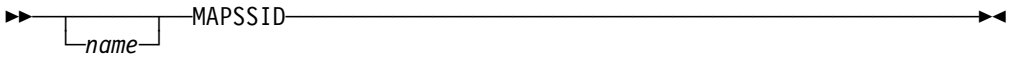
Figure 40. Layout of the STXIT Save Area (AMODE=24 and MSGDATA=NO)

If AMODE=ANY or MSGDATA=YES is specified in the STXIT macro, the **extended** save area as shown in Figure 41 on page 310 is used:

Field Name	No. of Bytes	Contents
SVUPSW	4	First half of (BC-mode) PSW of interrupted program
SVUPSW2	4	Second half of PSW
SVUR00	4	Save area for register 0
SVUR01	4	Save area for register 1
.		.
.		.
.		.
SVUR0F	4	Save area for register 15
SVUOLDLN		Old save area length
SVUAPSW	8	Actual PSW of interrupted program
	8	Reserved
	64	Exit-dependent area:
SVUABINF		AB exit: Cancel information
SVUMGADR		OC exit: Data from MSG command
SVUMCSID		4-byte console ID (CONSID) of the console where the MSG command was entered. The setting of the high-order bit indicates if the console has 'master' (0) or 'user' (1) authority.
SVUMNAME		8-byte name of the console where the MSG command was entered.
SVUMCART		8-byte command and response token (CART) associated with the MSG command.
SVUMDLNG		2-byte length of MSG data.
SVUMDATA		31-bit pointer to MSG data (zero if no data is specified).
SVUEXLNG		Length of OC exit extension
SVUAREG	64	Save area for access registers
SVUAR00		Save area for access register 1
.		.
.		.
.		.
SVUAR0F		Save area for access register 15
SVULNGTH		New save area length
SVUOCLEN		Save area for OC exit extension

Figure 41. Layout of the Extended STXIT Save Area (AMODE=ANY or MSGDATA=YES)

MAPSSID (Map for SUBSID) Macro



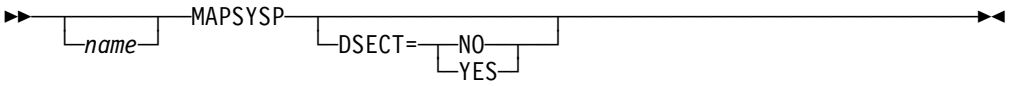
The macro generates a mapping DSECT which is used to interpret the supervisor information retrieved with the SUBSID macro. The output shows the supervisor identification string in the format as shown in Figure 42.

Field Name	No. of Bytes	Contents
IJBSSID1	2	Always zero
IJBSNAME	4	Character string: SUP
IJBSVERS	1	Version number *
IJBSREL	1	Release number
IJBSMOD	1	Modification level
IJBSVARL	1	Length of variable part (a max. of 24 bytes)
IJBSFL01	1	Flag byte 1:
IJBSF370		X'80' ESA-mode supervisor
IJBSFCKD		X'20' CKD support available
IJBSFFBA		X'10' FBA support available
IJBSFAPR		X'08' 3800 support available
IJBRCHAN		X'04' Relocating channels
IJBSVMLE		X'02' Reserved
IJBSVMAC		X'01' Any supervisor active under VM control
IJBSFL02	1	Flag byte 2:
IJBSFAF		X'80' Always 1
IJBSFPAG		X'40' 4K page size
IJBSUNAT		X'20' Running unattended
IJBSESAS		X'10' Reserved
IJBSACCR		X'08' Access registers available
IJBSFL03	1	Flag byte 3:
IJBSFSEC		X'80' Access-control support is available
IJBSFSHR		X'40' DASD sharing support is available
IJBSFSAT		X'20' For internal use
IJBSFL04	1	Flag byte 4: Reserved
IJBSLCON	2	Length of sublibrary search chain
IJBSFIXL		Length of fixed part
IJBSSLEN		Total length of supervisor entry

* Note that 'Version 6' refers to VSE/Advanced Functions Version 6, which actually stands for VSE/ESA Version 2.

Figure 42. Layout of MAPSSID-Generated DSECT

MAPSYSP (Map System Layout) Macro



Required RMODE: **24** or **ANY**

The macro may be used to interpret the information retrieved by the EXTRACT macro for ID=BDY and MODE=SYSP. If 'name' is omitted, MAPSYSP is taken as default.

DSECT=NO | YES

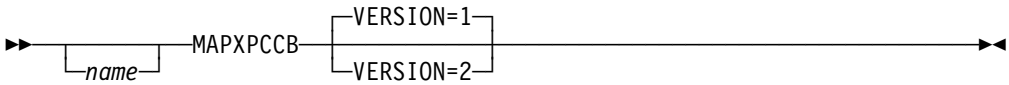
DSECT=YES specifies that a mapping DSECT is generated. If the operand is omitted, inline code is generated.

For the layout and a description of the DSECT fields, see Figure 43.

Field Name	No. of Bytes	Contents
MAPSUPBG	4	Begin of supervisor area
MAPSUPND	4	End of supervisor area
MAPSSBEG	4	Begin of SDAID area
MAPSSEND	4	End of SDAID area
MAPSVA	4	Begin of Shared Virtual Area
MAPESVA	4	End of Shared Virtual Area
MAPSPBEG	4	Begin of shared partition area
MAPSPEND	4	End of shared partition area
MAPPRPBG	4	Begin of private partition area
MAPPRPND	4	End of private partition area
MAPVPBEG	4	Begin of VPOOL area
MAPVPEND	4	End of VPOOL area
MAPSVIS	4	Begin of system GETVIS area
MAPSVISE	4	End of system GETVIS area
MAPPPBEG	4	Begin of problem program area (may be shared or private)
MAPSYLEN		Length of MAPSYSP area

Figure 43. Layout of the MAPSYSP-Generated DSECT

MAPXPCCB (Map Cross-Partition Control Block) Macro



This macro causes a DSECT of the cross-partition communication control block XPCCB to be generated.

VERSION=1 | 2

Indicates the version of the XPCCB. Two versions of the XPCCB are maintained. VERSION=1 is the default. VERSION=2 is required if the application wants to make use of the TIMEOUT, SENDI, or MECB functions of the XPCC support.

The most important fields of the XPCCB are described in the following figures:

- IJBXRETC (return codes) - see Figure 44 on page 314
- IJBXREAS (reason codes, set when an ECB is posted) - see Figure 45 on page 316
- IJBXFCT (function codes) - see Figure 46 on page 316
- IJBXFDSC (function descriptors) - see Figure 47 on page 317.

Reg. 15	IJBXRETC	(Symbolic Name)	Reason
X'00'	X'00'	(IJBXREOK)	Request completed successfully.
X'04'	X'01'	(IJBXDAPP)	Identification with the same application was requested previously in different partition. The connection is granted.
	X'02'	(IJBXAPSP)	Identification with the same application was requested previously in same partition. The connection is granted.
	X'03'	(IJBXFCRQ)	More than one CONNECT request is pending for this application.
	X'04'	(IJBXNIDN)	Other side did no IDENT until now.
	X'05'	(IJBXNCNN)	Other side did no CONNECT for this application until now.
	X'1B'	(IJBXOICL)	Request already cleared.
	X'20'	(IJBXSSWI)	SENDI protocol switched to SEND protocol because RECEIVE area is too small.
	X'21'	(IJBXWECB)	Invalid main ECB address (address ignored).
	X'22'	(IJBXBUFS)	CLEAR request accepted, wait on IJBXSECB for completion.
X'08'	X'06'	(IJBXWCBK)	XPCCB control block format error.
	X'07'	(IJBXWIDK)	Wrong identify token. (Token is invalid, or application issued TERMQSCE already, or CONNECT from pseudo partition without an IDENTify.)
	X'08'	(IJBXWPID)	Wrong path ID.
	X'09'	(IJBXWOWN)	Request was done under a task that has an incorrect task ID.
	X'0A'	(IJBXWIND)	Buffer list was specified for a function that does not support it.
	X'0B'	(IJBXWLST)	Too many buffers, or buffer length exceeds 16M bytes, or length is smaller than 2 in a CMS connection
	X'0C'	(IJBXWRAR)	Receiving buffer is too small.
	X'0D'		Reserved.
	X'0E'	(IJBXNSTO)	Try later, not sufficient storage to allocate system control blocks.
	X'0F'		Reserved.
	X'10'	(IJBXNREQ)	No request pending. (Line not busy, or SEND was from other side, or data already cleared).
	X'11'	(IJBXCCLR)	Request was already cleared.

Figure 44 (Part 1 of 2). MAPXPCCB Macro Return Codes (IJBXRETC)

Reg. 15	IJBXRETC	(Symbolic Name)	Reason
X'08'	X'12'	(IJBXCBSY)	Communication link already busy (FUNC=SEND/R).
	X'13'	(IJBXWSEQ)	REPLY was issued before data was received.
	X'14'	(IJBXNTRM)	At least one connection is still busy for the application.
	X'15'	(IJBXNDC1)	Busy from own SEND (FUNC=DISCONN).
	X'16'	(IJBXNDC2)	SEND from other side pending (FUNC=DISCONN).
	X'17'	(IJBXQSCE)	Other side issued TERMQSCE.
	X'18'	(IJBXNOC1)	A connection was never existing.
	X'19'	(IJBXNOC2)	The other side terminated normally
	X'1A'	(IJBXNOC3)	The other side terminated abnormally.
	X'1C'	(IJBXWCBA)	XPCCB address of this request differs from the one given with CONNECT.
	X'1D'	(IJBXER25)	Input contains invalid address.
	X'1E'	(IJBXWSIS)	Wrong sequence in SENDI protocol.
	X'1F'	(IJBXWTSK)	Task requested communication with itself.
	X'23'	(IJBXDUP)	Application name was specified twice, at least once with UNIQUE.
	X'24' to X'29'		Reserved.
	X'2A'	(IJBXINAM)	Invalid application name (application name contains a blank or all binary zeros).
	X'2B'	(IJBXTIMO)	A connection never existed because TIMEOUT occurred, or TIMEOUT=0 was specified with CONNECT, but the target application has not yet issued CONNECT.

Figure 44 (Part 2 of 2). MAPXPCCB Macro Return Codes (IJBXRETC)

Symbolic Name	Hex Value	Posted ECB	Description
One of the following reason codes will be set up in IJBXREAS:			
IJBXCPRG	X'01'	IJBXSECB	After SEND/SENDR the receiver issued PURGE.
IJBXCLEA	X'02'	IJBXRECB	Sender issued CLEAR before receiver was able to receive/reply.
IJBXRECX	X'03'	IJBXCECB	After SENDR command, RECEIVE is being executed by partner.
	X'04'		Reserved.
	X'05'		Reserved.
IJBXMQSD	X'06'	IJBXSECB IJBXRECB IJBXCECB	Maintask issued TERMQSCE and subtask has CONNECT requests open.
IJBXSWSR	X'07'	IJBXRECB	A SENDI was changed to SEND because RECEIVE area was too small.
	X'08' to X'0B'		Reserved.
IJBXTOUT	X'0C'	IJBXCECB	The specified time interval elapsed and the requested application has not issued CONNECT (DISCONNECT is required for acknowledgement).
The next two reason codes are OR'ed to the reason code field.			
IJBXDISC	X'40'	IJBXSECB IJBXRECB IJBXCECB	Other side issued DISCONNECT.
IJBXABDC	X'80'	IJBXSECB IJBXRECB IJBXCECB	Other side was disconnected due to abnormal termination.

Figure 45. MAPXPCCB Reason Codes (IJBXREAS)

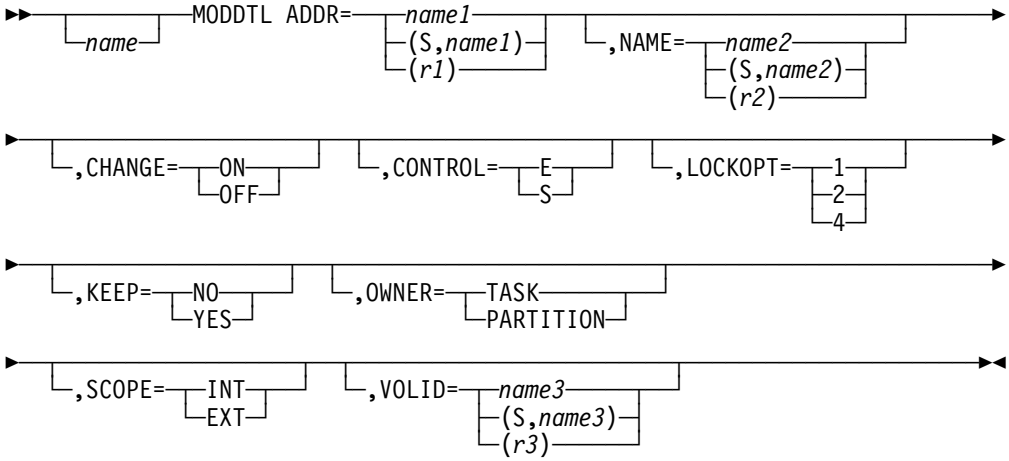
Symbolic Name	IJBXFCT Hex Val	Description
IJBXID	X'01'	Identify
IJBXCON	X'02'	Connect
IJBXSND	X'03'	Send
IJBXSNDR	X'04'	Send with reply
IJBXRCV	X'05'	Receive
IJBXREP	X'06'	Reply
IJBXCLR	X'07'	Clear
IJBXPRG	X'08'	Purge
IJBXDSC	X'09'	Disconnect
IJBXDSCP	X'0A'	Disconnect and purge
IJBXDSCA	X'0B'	Disconnect all
IJBXTRM	X'0C'	Terminate
IJBXTRMP	X'0D'	Terminate and purge
IJBXTRMQ	X'0E'	Terminate and quiesce
IJBXSNDI	X'0F'	Send with immediate move

Figure 46. MAPXPCCB Function Codes (IJBXFCT)

Symbolic Name	IJBXFDSC Hex Val.	Valid with	Description
IJBXPOST	X'80'	SENDR	The sender requests posting of IJBXCECB when the data has been received.
IJBXUNIQ	X'40'	IDENT	The current application name is requested to be unique.
IJBXFTIM	X'08'	CONNECT	Timeout required
IJBXFDAB	X'10'	DISCONN DISCPRG	The requested disconnect has to be indicated as abnormal.

Figure 47. MAPXPCCB Function Descriptor Codes (IJBXFDSC)

MODDTL (Modify DTL Block) Macro



Requirements for the caller:

AMODE: 24 or 31

RMODE: 24

ASC Mode: Primary

The macro modifies operands (fields) of a DTL (Define The Lock) control block. A DTL is used by the LOCK/UNLOCK macros to enqueue/dequeue a specific resource. The control block must have been generated by the DTL or GENDTL macro.

Operands not specified in the MODDTL macro leave the corresponding field in the DTL unchanged. There are no default values for the MODDTL macro.

ADDR=name1 | (S,name1) | (r1)
Specifies the address of the DTL.

NAME=name2 | (S,name2) | (r2)
Specifies the address of the area where a 12-byte long resource name is stored. If the name is shorter than 12 bytes, it must be padded with blanks. It is by this name, that VSE controls shared access of the resource as requested by active tasks via the LOCK macro. These tasks may all be

active in one partition, or they may be distributed over several partitions; the resource-share control extends across partitions.

The name you specify must not begin with any of the characters A through I or V because these characters are reserved for IBM.

CHANGE=ON | OFF
CHANGE=ON sets up the DTL such that a subsequent UNLOCK macro would not release the resource, but reduce its locking status. Reducing the lock status can be done only when the current lock status is defined with strongest possible values: CONTROL=E and LOCKOPT=1. At least one of the operands CONTROL and LOCKOPT should be specified, too. CHANGE=OFF causes a subsequent UNLOCK macro to resume its normal function: to dequeue the resource.

CONTROL=E | S
Defines how the named resource can be shared while your program owns it, which is determined by this specification and your specification for the operand LOCKOPT. A specification of E means the resource is enqueued for exclusive use; a specification of S means the resource is enqueued as sharable.

LOCKOPT=1 | 2 | 4
Together with the CONTROL operand, this operand determines how the system controls shared access in response to a LOCK request.

LOCKOPT=1 and CONTROL=E
No other task is allowed to use the resource concurrently.

LOCKOPT=1 and CONTROL=S
Other 'S' users are allowed concurrent access, but no concurrent 'E' user is allowed.

LOCKOPT=2 and CONTROL=E
No other 'E' user gets concurrent access; however, other 'S' users can have access to the resource.

LOCKOPT=2 and CONTROL=S
Other 'S' users can have concurrent access and, in addition, one 'E' user is allowed.

LOCKOPT=4 and CONTROL=E
No other 'E' user **from another system** is allowed concurrent access. However, other 'S' users from other systems may use the resource concurrently. Within your own system, you always have access to the resource.

LOCKOPT=4 and CONTROL=S
Other 'S' users can have concurrent access and, in addition, one 'E' user **from another system** is allowed.

Note: If the DASDSHR support is not generated in the supervisor, the LOCK request for the resource is always granted.

All users of a particular resource have to use the same LOCKOPT specification when they lock the resource. Exception: If LOCKOPT=1 and CONTROL=E, the lock status may be modified.

KEEP=NO | YES

This operand may be used to lock the named resource beyond job step boundaries. Only a main task should use this operand. KEEP=NO indicates that the named resource once locked, is to be released automatically at the end of the particular job step. With KEEP=YES, a named resource that is locked remains locked across job steps; it will be automatically released at end of job. If a job terminates abnormally, all resources with KEEP=YES are unlocked by the abnormal termination routine.

OWNER=TASK | PARTITION

Defines whether the named resource, once locked, can be unlocked only by the task which issued the corresponding LOCK request (OWNER=TASK), or whether it can be unlocked by any task within the partition (OWNER=PARTITION).

When OWNER is defined as PARTITION, a LOCK request for the resource must not specify FAIL=WAIT or FAIL=WAITC because deadlock prevention (return code 16) is not supported with OWNER=PARTITION.

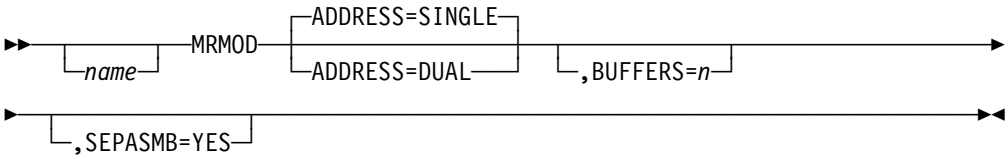
SCOPE=INT | EXT

This operand may be used for locking resources across systems. SCOPE=EXT specifies that the lock is used across systems. Specify SCOPE=INT if the locking is to apply to one system only.

VOLID=name3 | (S,name3) | (r3)

Specifies the address of a 6-byte identifier of a DASD volume which (at the time of the LOCK request) is to be checked whether it is mounted on an I/O device that is defined as being shared across systems. If the device is a shared DASD, the LOCK request is treated as being defined with SCOPE=EXT (that is, the SCOPE operand is ignored); otherwise, SCOPE=INT is assumed.

MRMOD (MICR Input Module Definition) Macro



Requirements for the caller:

AMODE: 24

RMODE: 24

ASC Mode: Primary

The macro defines a logic module for a MICR or OCR file. If you do not specify a name, IOCS assigns a standard module name: IJUxZZZZ. In this name:

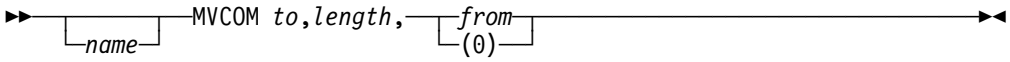
```
x = S: single address adapter
    D: dual address adapter
```

ADDRESS=SINGLE | DUAL
Required only if the dual address adapter is used for the IBM 1419 or 1275. If the operand is omitted, the single address adapter is assumed by the assembler.

BUFFERS=n
A numeric value equal to the corresponding value specified in the DTFMR macro.

SEPASMB=YES
Include this operand only if the module is to be assembled separately. This produces an object module ready to be cataloged into a suitable sublibrary either by the standard name or by the user-specified name. The name is used as the module's transfer address. If you omit the operand, the assembler assumes that the MRMOD macro is assembled together with the DTF in your program.

MVCOM (Move to Communication Region) Macro



Requirements for the caller:

AMODE: 24

RMODE: 24

ASC Mode: Primary

The MVCOM macro modifies the content of bytes 12 through 23 of the communication region of the partition from which the macro is issued. This area is commonly referred to as the user area. For the layout of the partition communication region, see “Communication via the Partition Communication Region” in the *VSE/ESA System Macro User's Guide*.

to Specifies the address (relative to the first byte of the region) of the first communication region byte to be modified.

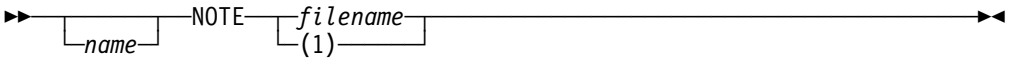
length
Specifies the number of bytes (1 to 12) to be inserted.

from | (0)
Specifies the address (either as a symbol or in register notation) of the bytes to be inserted.

The following example shows how to move three bytes from the symbolic location DATA into bytes 16 through 18 of the communication region:

```
MVCOM  16,3,DATA
```

NOTE (Note-Address) Macro



Requirements for the caller:

AMODE: 24

RMODE: 24

ASC Mode: Primary

filename | (1)

The macro obtains identification for a physical record or logical block that was last read from or written to the file specified for filename (as a symbol or in register notation). At least one READ or WRITE operation should be successfully completed by means of the CHECK macro before issuing the NOTE macro. To NOTE a desired record successfully, the POINTR, POINTS, or POINTW macros must not be issued between CHECK and NOTE.

For magnetic tape, the last record read or written in the specified file is identified by the number of physical records read or written from the load point. The physical record number is returned in binary in the three low-order bytes of register 1. The high-order byte contains binary zero.

For CKD DASD, the binary number returned in register 1 is in the form cchr, where

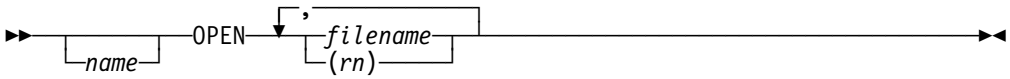
```
cc = cylinder number
 h = track number
 r = record number within the track
```

Register 0 contains, in the two low-order bytes, the unused space remaining on the track following the end of the identified record.

For FBA devices, register 1 contains an address relative to the beginning of the file in the form cccb, where ccc is the relative number of the current control interval (origin 0), and b is the relative block number within the current CI (origin 1). Register 0 contains the length of the longest logical block that could completely fit in the CI following the NOTEd logical block. A logical block three bytes longer than the returned value will fit in the CI if it is of the same length as both the NOTEd block and the block preceding the NOTEd block. (This means that if the CI were exactly filled when the NOTE was issued, a value of -3 would be passed back in register 0.)

You must provide a four- or six-byte field and store in it the record identification and the remaining capacity so that it can be used later by a POINTR or POINTW macro to find the NOTEd record again. The remaining two-byte track or CI capacity is needed only when a WRITE SQ is to follow the POINTR or POINTW.

OPEN and OPENR (Open a File) Macro



Requirements for the caller:

AMODE: 24 or 31

RMODE: 24

ASC Mode: Primary

Except for the code in the operation field, the format of the OPENR macro is the same as that of the OPEN macro.

The OPEN (or OPENR) macro activates all files.

When OPENR is specified, the symbolic address constants generated from the operands of the macro are self-relocating. When OPEN is specified, the symbolic address constants are not self-relocating. Throughout the manual the term OPEN also implies OPENR, unless stated otherwise.

OPEN need not be issued for DTFCN files in a non-selfrelocating environment. However, self-relocating programs using LIOCS must specify OPENR for all files, including console files.

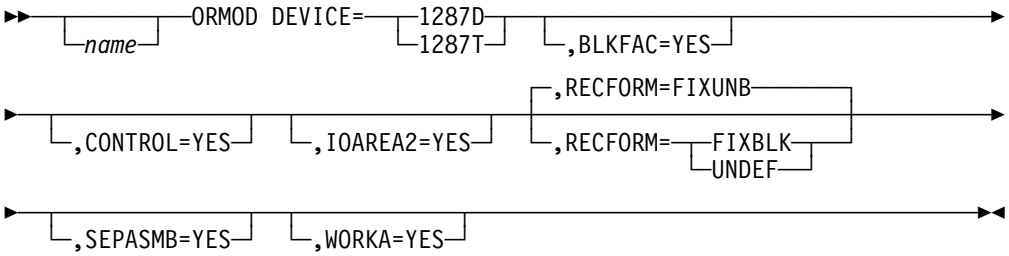
If OPEN attempts to activate a file whose device is unassigned, the job is terminated. If the device is assigned IGN, OPEN does not activate the file, but turns on bit 2 of the DTF byte 16, which indicates that the file is not activated. If this bit is on after issuing an OPEN, I/O operations may not be attempted for the file.

filename | (rn)

Code the symbolic name of the file (DTF filename) to be opened. You can open up to 16 files with one macro by coding additional file names. Alternatively, you can load the address of a file name into a register and specify the register, using ordinary register notation.

The high-order 8 bits of this register must be zeros. For OPENR, the address of the file's name may be preloaded into any of the registers 2 through 12. For OPEN, this address may be preloaded into register 0 or any of the registers 2 through 12.

ORMOD (Optical Reader Input Module Definition) Macro



Requirements for the caller:

AMODE: 24

RMODE: 24

ASC Mode: Primary

The ORMOD macro defines a logic module for an IBM 1287 or 1288 optical reader file.

Note: ORMOD is not used for the 3881 Optical Mark Reader. The IBM 3881 uses CDMOD.

BLKFAC=YES
Include this operand if RECFORM=UNDEF and groups of undefined journal tape records are to be processed as blocks of data. For more information, see the DTFOR BLKFAC=n operand. The DTFOR used with this module must include RECFORM=UNDEF and BLKFAC=n.

CONTROL=YES
Include this operand if CNTRL macros are to be used with the associated DTFs. The module also processes files that do not use the CNTRL macro.

DEVICE=1287D | 1287T
This operand must be included to specify the I/O device associated with this file. 1287D specifies a 1287 or 1288 document file. 1287T specifies a 1287 journal tape file.

IOAREA2=YES
Include this operand (journal tape only) if a second I/O area is used. The DTFOR used with this module must also include the IOAREA2 operand.

RECFORM=FIXUNB | FIXBLK | UNDEF
This operand generates a module that processes the specified record format. Any DTF used with the module must have the same operand.

SEPASMB=YES
Include this operand if the module is to be assembled separately. This produces an object module ready to be cataloged into a suitable sublibrary, either by the standard name or by the user-specified name. This name is used also as the module' transfer address. If you omit this operand, the assembler assumes that the module is assembled together with the DTF in your program.

WORKA=YES

Include this operand (journal tape only) if records are to be processed in work areas instead of in I/O areas. Any DTF used with the module must have the same operand.

Standard ORMOD Names

Each name begins with a 3-character prefix (IJM) followed by a 5-character field corresponding to the options permitted in the generation of the module.

ORMOD name = IJMabcde

Char.	Content	Specified Option
a	F	RECFORM=FIXUNB
	X	RECFORM=FIXBLK
	U	RECFORM=UNDEF
	D	RECFORM=UNDEF and BLKFAC=YES
b	C	CONTROL=YES
	Z	CONTROL=YES is not specified
c	I	IOAREA2=YES
	W	WORKA=YES
	B	Both are specified
	Z	Neither is specified
d	T	Device is in tape mode
	D	Device is in document mode
e	Z	Always

Subset/Superset ORMOD Names

The following chart shows the subsetting and supersetting allowed for ORMOD names. One of the operands allows subsetting. For example, the module IJMFCITZ is a superset of the module IJMFZITZ.

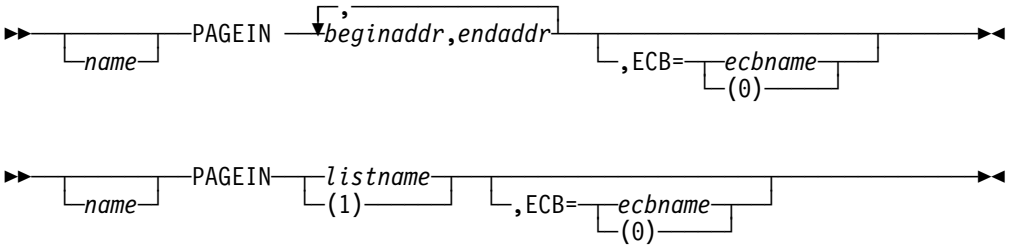
```
              *  +  *  *
       I  J  M  D  C  B  D  Z
                F  Z  I  T
                U     W
                X     Z
```

+
Subsetting/supersetting permitted

***** No subsetting/supersetting permitted

PAGEIN (Page-In Request) Macro

You can code the macro in either of the following formats:



Requirements for the caller:

AMODE: 24 (if SPLEVEL SET=1)

24 or 31 (if SPLEVEL SET>1)

RMODE: 24 (if SPLEVEL SET=1)

24 or ANY (if SPLEVEL SET>1)

ASC Mode: Primary

The macro causes specific areas to be brought into real storage before their contents are needed by the requesting program. If the requested area is already in real storage the attached page frame will get low priority for the next page-outs. This function, however, does not include any fixing, so that it cannot determine whether all areas requested will still be in real storage when the entire request has been completed.

The system can handle up to 15 active PAGEIN requests at any point in time.

In a system without page data set, execution of the macro results in a null operation. If the ECB operand is specified, the system posts the specified event control block.

beginaddr
Points to the first byte of the area to be paged in.

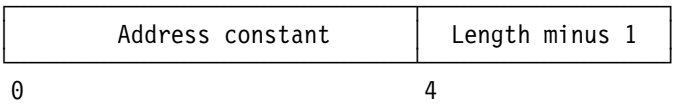
endaddr
Points to the last byte of the area to be paged in.

listname | (1)
Is the name of a list of consecutive 8-byte entries as shown below. Register notation may also be used. The address of this parameter list and the addresses in the list are treated as 3-byte addresses if the macro is invoked in 24-bit addressing mode and as 4-byte addresses if invoked in 31-bit addressing mode.

24-bit Addressing Mode:

X'00'	Address constant	Length minus 1
0	1	4

31-bit Addressing Mode:



where:

Address constant =
Address of the first byte of the area to be paged in.

Length = A binary constant indicating the length of the area to be paged in.

The end of the list is indicated by a non-zero byte following the last entry (for 24-bit addressing mode). For 31-bit addressing mode, a non-zero value in bit 0 of the byte following the last entry indicates the end of the list.

ECB=ecbname | (0)
Specifies the symbolic address of the ECB, a fullword defined by your program, which is to be posted when the operation is complete. An invalid ECB address causes the task to be canceled.

Return Information

The return information can be obtained from byte 2 of the ECB. The meaning of these bits is shown below.

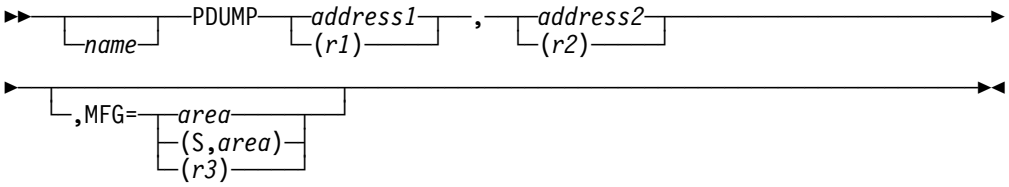
Bit	Meaning if the Bit is One
0	PAGEIN request is finished.
1	The page table is full, the request cannot be queued at this time for further handling; the request is ignored, bit 0 is set.
2	One or more of the requested pages are outside the requesting program's partition; PAGEIN is not performed for these pages.
3	At least one negative length has been detected in the area specifications; PAGEIN is not performed for these areas.
4	List of areas that are to be paged in is not completely in the requesting program's partition; the request is ignored, bit 0 is set.
5	Paging activity is too high in the system, no performance improvement is possible; the request is terminated, bit 0 is set.
6	Reserved.
7	Inconsistent function/option code provided in register 15 (not possible if macro interface is used).

Any combination of the return bits in the ECB is possible.

Use the WAIT macro with the ecbname as operand for completion of the PAGEIN macro, before the bits in byte 2 of the ECB are tested.

The PAGEIN function runs asynchronously with the requesting user task; therefore, if no ECB has been specified, the requesting task cannot be notified when the PAGEIN function is completed.

PDUMP (Partial-Dump Request) Macro



Requirements for the caller:

AMODE: 24 or 31

RMODE: 24

ASC Mode: Primary

This macro provides a hexadecimal dump of the general registers and of the virtual storage area contained between the two address expressions (address1 and address2). The contents of registers 0 and 1 are overwritten, but the CPU status is retained. Thus, PDUMP furnishes a dynamic dump (snapshot) useful for program checkout. Processing continues with your next instruction.

Restrictions: The PDUMP macro cannot be used to process 31-bit addresses; these addresses must be specified in register notation. Also, the PDUMP macro cannot be used to dump data spaces; use the SDUMP or SDUMPX macro for this purpose.

The dump is always directed to SYSLST with 121-byte records (the first byte is an ASA control character). If SYSLST is not assigned, the PDUMP macro is ignored. If SYSLST is assigned to a CKD-type disk device, no output will be produced.

If SYSLST is assigned to an IBM 3211 and indexing was used before you issue the PDUMP macro, a certain number of characters on every line of the printed dump may be lost. To avoid this, reload the printer's FCB (forms control buffer) by issuing an LFCB macro before you issue the DUMP macro. The FCB image you load must not have an indexing byte.

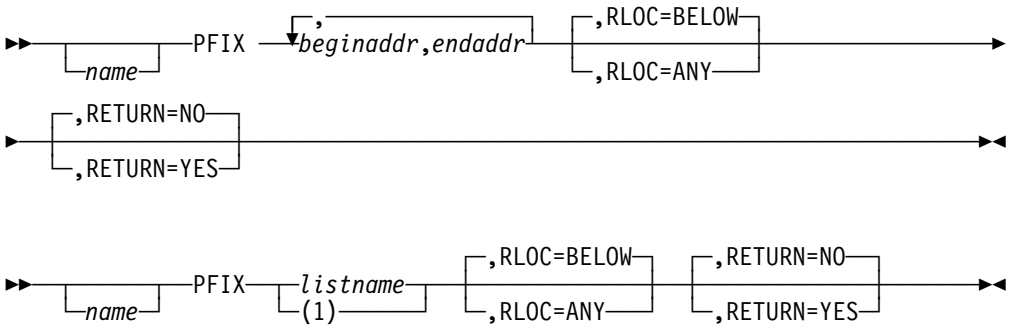
If non-addressable areas are included in the range of PDUMP, the system issues a message to indicate this.

address1 | (r1),address2 | (r2)
: One or both of the addresses can be specified in register notation. The specified addresses must be 3-byte addresses; 31-bit addresses are only allowed in register notation. If address2 is not greater than address1, or address1 is greater than the highest address in the allocated virtual storage, the macro results in no operation. If the value in address2 is greater than the end of the allocated virtual storage area, the virtual storage between address1 and the end of the allocated virtual storage is dumped.

MFG=area | (S,area) | (r3)
: The operand is required if the program which issues the PDUMP is to be reenterable. It specifies the address of a 64-byte dynamic storage area, that is, storage which you obtained by a GETVIS macro; this area is needed by the system during execution of the macro.

PFIX (Page-Fix Request) Macro

You can code the macro in either of the following formats: with an explicitly provided fix list or, in the generation format, with the implicitly generated fix list in the macro expansion:



Requirements for the caller:

AMODE: 24 (if SPLEVEL SET=1) 24 or 31 (if SPLEVEL SET>1)

RMODE: 24 (if SPLEVEL SET=1) 24 or ANY (if SPLEVEL SET>1)

ASC Mode: Primary

The macro causes specific pages to be brought into real storage and fixed in their page frames until they are released at some later time. The maximum number of pages that may be fixed at any one time is specified via the ALLOC R or the SETPFIX job control command. Each time a page is fixed, a counter for that page is incremented. This counter must not exceed 32,767 for any page.

beginaddr
Points to the first byte of the area to be fixed.

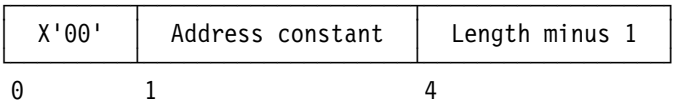
endaddr
Points to the last byte of the area to be fixed.

RLOC=BELOW | ANY
Specifies the location of the real storage for the PFIX requests: BELOW indicates below 16MB, ANY indicates anywhere. (For ANY, the system first tries to PFIX in the area above the 16MB line and only if this area is already totally PFIXed, it tries to PFIX in the area below the 16MB line.)

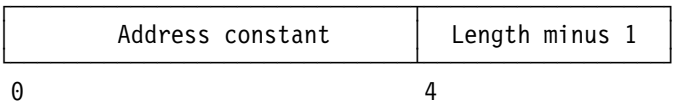
RETURN=NO | YES
YES indicates that control is to be returned to the issuer of the PFIX if the request cannot be satisfied because of temporarily fixed pages. NO indicates that the issuer of PFIX has to wait till the requested pages have been fixed.

listname | (1)
Is the name of a list of consecutive 8-byte entries as shown below. (Register notation may also be used.) The address of this parameter list and the addresses in the list are treated as 3-byte addresses if the macro is invoked in 24-bit addressing mode and as 4-byte addresses if invoked in 31-bit addressing mode.

24-bit Addressing Mode:



31-bit Addressing Mode:



where:

Address constant =
Address of the first byte of the area to be fixed.

Length = A binary constant indicating the length of the area to be fixed.

The end of the list is indicated by a non-zero byte following the last entry (for 24-bit addressing mode). For 31-bit addressing mode, a non-zero value in bit 0 of the byte following the last entry indicates the end of the list.

Exceptional Conditions

- If a PFIX causes the maximum count of fixes for a page to be exceeded, the task issuing the PFIX is canceled.
- If it is not possible to fix all pages requested, then none will be fixed.
- If PFIX is issued in a program running in real mode, it is ignored and register 15 contains 0.

Return Codes in Register 15

0 The pages were successfully fixed.

4 The number of pages to be fixed for one request exceeds the number of PFIXable page frames; in order for this PFIX request to be satisfied, more PFIXable storage must be allocated through the ALLOC R or SETPFIX job control command.

8 Not enough page frames are available in the partition because of previous PFIXes or current system resource usage; this PFIX request could, however, be satisfied at another time without reallocating PFIXable storage.

12 One of the specified addresses was invalid, or begin address was higher than end address, or a negative length was found.

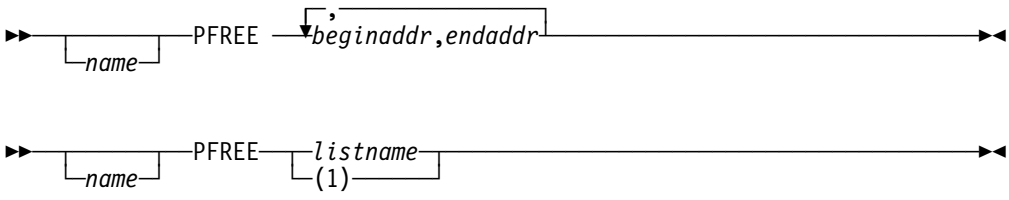
16 A PFIX request was given with RLOC=BELOW, but at least one page of the requested area is already PFIXed in a frame above 16MB by a previous request. The request is ignored, no page is fixed. A subsequent PFIX request with the same list and RLOC=ANY does, however, PFIX the area.

20 Inconsistent function/option code provided in register 15; no page is fixed (not possible if macro interface is used).

24 No pages were PFIXed because not enough page frames were available due to temporarily fixed pages.

PFREE (Page-Free Request) Macro

You can code the macro in either of the following formats: with an explicitly provided fix list or, in the generation format, with the implicitly generated fix list in the macro expansion:



Requirements for the caller:

AMODE: 24 (if SPLEVEL SET=1)

24 or 31 (if SPLEVEL SET>1)

RMODE: 24 (if SPLEVEL SET=1)

24 or ANY (if SPLEVEL SET>1)

ASC Mode: Primary

The macro frees one or more pages previously PFIXed in real storage.

Each page in the virtual address area is assigned a 'PFIX counter'. If a page is not fixed - that is, if it is subject to normal page management - the counter is 0. Whenever a page is fixed by using a PFIX macro its counter is increased by one. All pages whose counters are greater than 0 remain fixed in real storage.

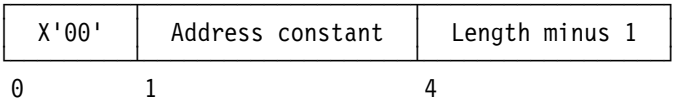
The PFREE macro decrements the counter of a specified page by 1. If a PFREE is issued for a page whose counter is 0, that PFREE is ignored.

beginaddr
Points to the first byte of the area to be freed.

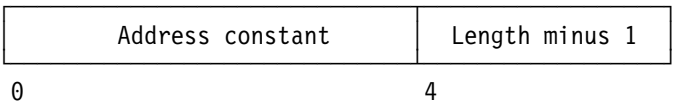
endaddr
Points to the last byte of the area to be freed.

listname | (1)
Is the name of a list of consecutive 8-byte entries as shown below. (Register notation may also be used.) The address of this parameter list and the addresses in the list are treated as 3-byte addresses if the macro is invoked in 24-bit mode and as 4-byte addresses if invoked in 31-bit mode.

24-bit Mode:



31-bit Mode:



where:

Address constant =
Address of the first byte of the area to be freed.

Length = A binary constant indicating the length of the area to be freed.

The end of the list is indicated by a non-zero byte following the last entry (for 24-bit mode). For 31-bit mode, a non-zero value in bit 0 of the byte following the last entry indicates the end of the list.

Exceptional Conditions

If PFREE is issued by a program running in real mode, the macro is ignored.

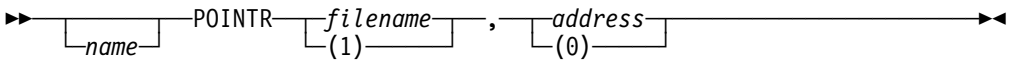
Return Codes in Register 15

0 The pages were successfully freed.

12 One of the specified addresses was invalid, or begin address was higher than end address, or a negative length was specified.

20 Inconsistent function/option code provided in register 15; no page is freed (not possible if macro interface is used).

POINTR (Point to Noted Record) Macro



Requirements for the caller:

AMODE: 24

RMODE: 24

ASC Mode: Primary

The macro repositions the file specified by filename to the record identified by previously issuing a NOTE macro.

If a READ follows the POINTR, the record noted by NOTE is the record read (tape or DASD).

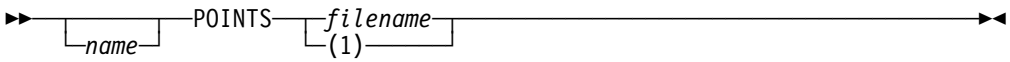
For magnetic tape, a WRITE must not follow a POINTR.

For a work file on disk, if a WRITE UPDATE follows the POINTR, the noted record is written (or overwritten). If a WRITE SQ follows the POINTR, the record **after** the noted one is written (or overwritten) and, on CKD DASD, the remainder of the track is erased (overwritten with zeros). On an FBA disk, the remainder of the CI is erased (overwritten with zeros) and an SEOF is written (the following CI is also overwritten with zeros).

filename | (1)
The filename may be expressed either as a symbol or in register notation.

address | (0)
Specifies the virtual storage location of either a four-byte record identifier or a four-byte record identifier followed by a two-byte track or CI capacity. The four- or six-byte number must be in the form obtained from the NOTE macro. The two-byte track or CI capacity is required only when a WRITE SQ is to be issued following the POINTR.

POINTS (Point to Start) Macro



Requirements for the caller:

AMODE: 24

RMODE: 24

ASC Mode: Primary

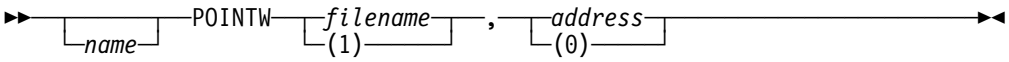
The POINTS macro repositions a file to its beginning.

For a tape file, the tape is rewound. If the file contains any header labels, they are bypassed, and the tape is positioned to the first record following the label set.

For disk work files, the file is repositioned to the lower limit of the first extent. A POINTS should not be followed by a WRITE UPDATE. If a POINTS is followed by a WRITE SQ, the first record in the file is overwritten. For CKD DASD, the remainder of the track is then erased (overwritten with zeros). For FBA devices, the remainder of the CI is erased (overwritten with zeros) and an SEOF is written (the following CI is also overwritten with zeros).

filename | (1)
The file name may be expressed either as a symbol or in register notation.

POINTW (Point Behind Noted Record) Macro



Requirements for the caller:

AMODE: 24

RMODE: 24

ASC Mode: Primary

The POINTW macro repositions the file specified by filename to the record following the record identified by previously issuing a NOTE macro. A READ or WRITE following a POINTW macro results in the following:

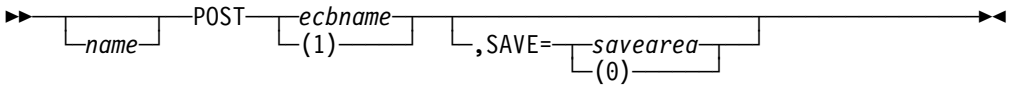
- For magnetic tape, a READ following a POINTW causes IOCS to read the record after the one noted by a NOTE macro.
- For DASD work files, a READ following a POINTW causes the noted record to be read.
- For magnetic tape, a WRITE UPDATE following a POINTW causes the record following the noted record to be overwritten.
- For a work file on disk, a WRITE UPDATE following a POINTW causes the noted record to be overwritten.

If a WRITE SQ follows the POINTW, the record **after** the noted one is written (or overwritten) and, on CKD disk, the remainder of the track is erased (overwritten with zeros). On an FBA disk, the remainder of the CI is erased (overwritten with zeros) and an SEOF is written (the following CI is also overwritten with zeros).

filename | (1)
The file name may be expressed either as a symbol or in register notation.

address | (0)
Specifies the virtual storage location of either a four-byte record identifier or a four-byte record identifier plus a two-byte track or CI capacity. The four- or six-byte number must be in the form obtained from the NOTE macro. The two-byte track or CI capacity is required only when a WRITE SQ is to be issued following the POINTW.

POST (Post Event) Macro



Requirements for the caller:

AMODE: 24 or 31

RMODE: 24 or ANY

ASC Mode: Primary

This macro provides communication between two tasks in the same partition by posting an event control block (ECB). It causes bit 0 of byte 2 of the ECB to be set on. A post issued to an ECB removes a task waiting for the ECB from the wait state. For more details about an ECB, see the description of the ECB operand of the ATTACH macro on page 26.

POST processing can post an ECB in 24-bit or 31-bit addressing mode physically resident above or below 16MB. When POST is issued in AMODE 24, all operands are treated as 24-bit addresses. When POST is issued in AMODE 31, all operands are treated as 31-bit addresses.

ecbname | (1)
Provides the address of the ECB that is to be posted.

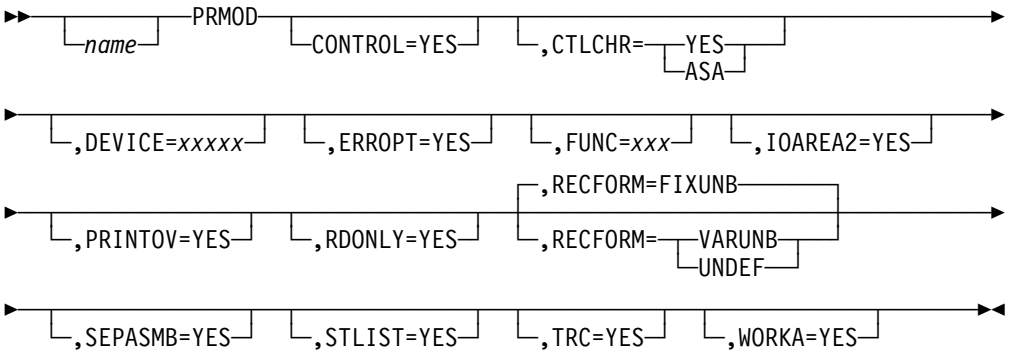
SAVE=savearea | (0)
This operand may be used for taking a specific waiting task out of the wait state. The operand causes the system to locate the save area whose address is specified in the operand and to take only the subtask associated with this save area out of the wait state. This task normally is waiting for the specified ECB to be posted. The save area has to be allocated below the 16MB line (RMODE 24).

Although specifying this operand saves time, since other tasks waiting for this ECB are not taken out of the wait state for the event that your program signals by issuing the POST macro, specifying the operand does not guarantee that those tasks will stay in the wait state until another POST is issued. Other events could cause the other tasks to be dispatched. Therefore, use the POST macro with SAVE to control subtask operation only if either:

- You have a separate ECB for each of the tasks, or
- There is a need to save processing time.

If you issue a POST without SAVE, all tasks waiting for the ECB are taken out of the wait state.

PRMOD (Printer Output Module Definition) Macro



Requirements for the caller:

AMODE: 24

RMODE: 24

ASC Mode: Primary

The macro defines a logic module for a printer file.

For printers of the type PRT1, 3800, and 4248 in native mode, the PRMOD macro is not needed. A logic module supplied for a printer of these types is ignored. OPEN always provides linkage to an IBM-supplied logic module. PRMOD is required for a 1403 and a 3525 printer.

CONTROL=YES
Include this operand if CNTRL macros are used with the associated DTFs. The module also processes files that do not use the CNTRL macro. If CONTROL is specified, the CTLCHR operand must not be specified.

CTLCHR=YES | ASA
Include this operand if first-character carriage control is used. Any DTF used with the module must have the same operand. If CTLCHR is specified, CONTROL must not be specified.

If CTLCHR=ASA is specified for a file on the IBM 3525, the + character is not allowed. For a 3525 print (not associated) file, you must issue either a space 1 command or skip to channel 1 command to print on the first line of a card. For 3525 print associated files, you must issue a space 1 command to print on the first line of a card.

Overprinting may occur if several DTFPRs of different tasks address the same device and at least one DTF specifies CTLCHR=ASA. Therefore, while a DTFPR (with CTLCHR=ASA) is doing an I/O operation, no other DTFPR should be allowed to do I/O on the same device.

DEVICE=xxxxx
This operand specifies which device is used for the file. For xxxxx, you can specify one of the following IBM device-type codes:

```
1403
3203
3211
3525
```

A device code of 3800 and PRT1 is accepted for compatibility reasons. The logic module is generated, but ignored if the DTF is opened.

Any DTF to be used with this module must have the same operand.

ERROPT=YES
The operand is valid only together with DEVICE=PRT1 and if the DTFPR macro for the file includes ERROPT=name. The operand is accepted for compatibility reasons.

FUNC=W | WT | RW | RWT | RPW | RPWT | PW | PWT
This operand specifies the type of file to be processed on your IBM 3525. Any DTF used with the module must include the same operand. W indicates print, R indicates read, P indicates punch, and T (for the IBM 3525 only) indicates an optional 2-line printer.

RW | RWT, RPW| RPWT, and PW | PWT are used to specify associated files. When one of these specifications is used for a printer file, this specification must be used also for the associated file(s).

If a 2-line printer is not specified for a file on the IBM 3525, multi-line print is assumed. T is ignored if CONTROL or CTLCHR is specified.

IOAREA2=YES
Include this operand if a second I/O area is used. Any DTF used with the module must also include the IOAREA2 operand.

PRINTOV=YES
Include this operand if PRTOV macros are used with the associated DTFs. The module also processes any files that do not use the PRTOV macro.

RDONLY=YES:
This operand causes a read-only module to be generated. Whenever this operand is specified, any DTF used with the module must have the same operand.

RECFORM=FIXUNB | VARUNB | UNDEF
This operand causes a module to be generated that processes the specified record format: fixed-length, variable-length, or undefined. Any DTF used with the module must include the same operand.

SEPASMB=YES
Include this operand only if the module is to be assembled separately. This produces an object module ready to be cataloged into a suitable sublibrary, either by the standard name or a name specified by you. The module name will be used also as the module entry-point. If you omit this operand, the assembler assumes that the module is assembled together with the DTF in your program.

STLIST=YES
Include this operand if the selective tape listing feature (IBM 1403 only) is used. If this entry is specified, the CONTROL, CTLCHR, and PRINTOV entries are not valid, and are ignored if supplied. If this operand is specified, RECFORM must specify FIXUNB.

TRC=YES

Include this operand to specify whether the module is to test the TRC bit in the DTFPR or ignore that bit. If TRC=YES is specified, the generated module can process output files with table reference characters and those without.

WORKA=YES

Include this operand if records are processed in work areas instead of in I/O areas. Any DTF used with the module must have the same operand.

Standard PRMOD Names

Each name begins with a 3-character prefix (IJD) followed by a 5-character field corresponding to the options permitted in the generation of the module.

PRMOD name = IJDabcde

Char.	Content	Specified Option
a	F	RECFORM=FIXUNB
	V	RECFORM=VARUNB
	U	RECFORM=UNDEF
b	A	CTLCHR=ASA
	Y	CTLCHR=YES
	C	CONTROL=YES
	S	STLIST=YES
	Z	None of these is specified
	T	DEVICE=3525 with 2-line printer
c	B	ERROPT=YES and PRINTOV=YES
	P	PRINTOV=YES, DEVICE is not 3525, and ERROPT is not specified
	I	PRINTOV=YES, DEVICE=3525, and FUNC=W|WT or omitted
	F	PRINTOV=YES, DEVICE=3525, and FUNC=RW|RWT
	C	PRINTOV=YES, DEVICE=3525, and FUNC=PW|PWT
	D	PRINTOV=YES, DEVICE=3525, and FUNC=RPW|RPWT
	Z	Neither PRINTOV nor ERROPT is specified, and DEVICE is not a 3525
	O	PRINTOV=YES not specified, DEVICE=3525, and FUNC=W|WT or omitted
	R	PRINTOV=YES not specified, DEVICE=3525, and FUNC=RW|RWT
	S	PRINTOV=YES not specified, DEVICE=3525, and FUNC=PW|PWT
	T	PRINTOV=YES not specified, DEVICE=3525, and FUNC=RP|RPT
	E	ERROPT=YES and PRINTOV=YES is not specified
d	I	IOAREA2=YES
	Z	IOAREA2=YES is not specified
e	V	RDONLY=YES and WORKA=YES
	W	WORKA=YES
	Y	RDONLY=YES
	Z	Neither is specified

Subset/Superset PRMOD Names

Two of the operands allow subsetting. For example, the module name IJDFCPIW is a superset of the module names IJDFCZIW and IJDFZZIW.

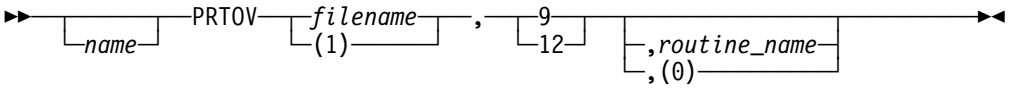
Note: The IBM-supplied modules were assembled without TRC=YES. You can reassemble them with TRC=YES to support 3800 table reference characters. Although the code generated for a module assembled with TRC=YES is different from the code generated for a module with TRC=NO, the module name is the same. If some, but not all PRMOD logic modules are reassembled this way, subsetting or supersetting may not work.

			*	*	+	*	*
I	J	D	F	A	P	I	V
			V	Y	Z	Z	W
			U	S	+		Y
				T	I		Z
				+	O		
				C	+		
				Z	F		
					R		
					+		
					+		
					C		
					S		
					+		
					D		
					T		
					+		
					B		
					E		

+
Subsetting/supersetting permitted.

***** No subsetting/supersetting permitted.

PRTOV (Printer Overflow Control) Macro



Requirements for the caller:

AMODE: 24

RMODE: 24

ASC Mode: Primary

The macro is used with a printer file to specify the operation to be performed when a carriage overflow condition occurs during the last PUT. To use this macro, the PRINTOV=YES operand must be included in the DTFPR macro.

filename | (1)
This operand is required. It must be the same as the name of the DTFPR macro for the printer file. You can code the operand as a symbol or in register notation.

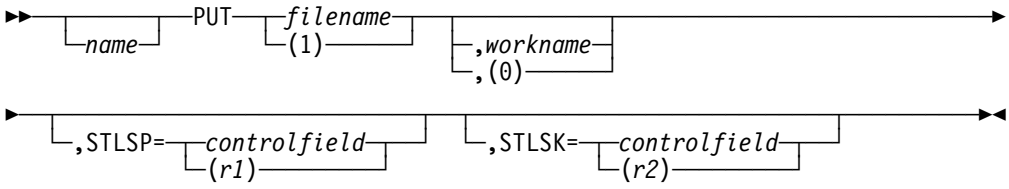
9 | 12 This operand is required. It specifies the number of the carriage control channel (9 or 12) used to indicate the overflow. When an overflow condition occurs, IOCS advances the printer carriage to the first printing line on the form (channel 1), and normal printing continues.

routine-name | (0)
Specify this operand only if you prefer to branch to your own routine on an overflow condition, rather than skipping directly to channel 1. It specifies the name of the routine, as a symbol or in register notation. However, the name should never be preloaded into register 1.

If you specify this operand, IOCS does not advance the carriage to channel 1.

Return from the overflow routine via register 14.

PUT (Put Record) Macro



Requirements for the caller:

AMODE: 24

RMODE: 24

ASC Mode: Primary

PUT writes, prints, or punches logical records which are built directly in the output area or in a specified work area. PUT can be used for any sequential output file defined by a DTF macro, and for any type of record. It operates much the same as GET but in reverse. It is issued after a record has been built.

filename | (1)
This operand must be the same as the name of the DTF macro for the file that is being built. The operand can be specified as a symbol or in either special or ordinary register notation. The high-order eight bits of the register must be zero. Use register notation if your program is to be self-relocating.

workname | (0)
This operand specifies a work area name or a register (in either special or ordinary register notation) containing the address of the work area. The work area address should never be preloaded into register 1.

This operand is used if the corresponding DTF contains the WORKA=YES operand indicating that the records are to be built in a work area which you define yourself (for example, using a DS instruction). If the operand is specified, all PUTs for the named file must use either a register or a workname. PUT then moves each record from the work area to the output area.

Individual records for a logical file may be built in the same work area or in different work areas. Each PUT macro specifies the work area where the completed record was built. However, only one work area can be specified in any one PUT macro.

Whenever a PUT macro transfers an output data record from an output area (or work area) to an I/O device, the data remains in the area until it is either cleared or replaced by other data. IOCS does not clear the area. Therefore, if you plan to build another record whose data does not use every position of the output area or work area, clear that area before you build the record. If this is not done, the new record will contain interspersed characters from the preceding record.

STLSP=controlfield | (r1)
This operand specifies a control byte that allows for spacing while using the selective tape listing feature on an IBM 1403. To use this feature, the operand STLST=YES must be specified in the DTFPR.

Up to 8 paper tapes may be independently spaced. The control byte is set up like any other data byte in virtual storage. You can also use ordinary register notation to provide the address of the control byte. Registers 2 through 12 are available without restriction. You determine the spacing (which occurs after printing) by setting on the bits corresponding to the tapes you want to space. The correspondence between control byte bits and tapes is as follows:

Bit	Tape Position
0	8 (rightmost tape)
1	7
2	6
3	5
4	4
5	3
6	2
7	1 (leftmost tape)

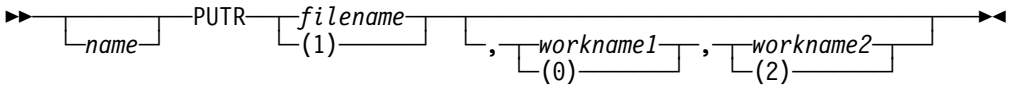
Note: Double-width tapes must be controlled by both bits of the control field.

STLSK=controlfield | (r2)

This operand specifies a control byte that allows for skipping while using the selective tape listing feature on the 1403 printer. To use this feature, the operand STLIST=YES must be specified in the DTFPR. Up to 8 paper tapes may be independently skipped. The control byte is set up like any other data byte in virtual storage. You can also use ordinary register notation to provide the address of the control byte. Registers 2 through 12 are available without restriction.

You determine the skipping (which occurs after printing) by setting on the bits corresponding to the tapes you want to skip. The correspondence between control byte bits and tapes is shown in the figure under "STLSP=controlfield", above.

PUTR (PUT with Reply) Macro



Requirements for the caller:

AMODE: 24

RMODE: 24

ASC Mode: Primary

The macro is used if a message is to be displayed at the system console and this message requires a reply. The message is not deleted from the display screen until the operator has issued the reply.

You may use PUTR also with the IBM 3215 console printer-keyboard, in which case PUTR functions the same as PUT followed by GET for these devices, but provides the message non-deletion code for the display operator console. Use of PUTR for the IBM 3215 is therefore recommended for compatibility if your program may at some time be run on a computer system with a display operator console.

Use PUTR for fixed unblocked records (messages). Issue PUTR after the record has been built.

Do not use register 2 as base register in any of the PUTR operands.

filename | (1)
This operand must be the same as the name of the DTFCN for the file that is being built. The filename can be specified as a symbol or in either special or ordinary register notation. The latter is necessary to make your programs self-relocating.

workname1 | (0)
This operand specifies the output work area name or a register (in either special or ordinary register notation) containing the address of the output work area. The work area address should never be preloaded into registers 1 or 2. This operand is used if records are built in a work area which you define yourself (for example, using a DS instruction). The length of the work area is defined by the BLKSIZE operand of the DTFCN macro. If workname1 is specified, workname2 must also be specified.

workname2 | (2)
This operand specifies the input work area name or a register (in either special or ordinary register notation) containing the address of the input work area. The work area address should never be preloaded into registers 0 or 1. The operand is used if records are built in a work area which you define yourself (for example, using a DS instruction). The length of the work area is defined by the INPSIZE operand of the DTFCN macro. If workname2 is specified, workname1 must also be specified.

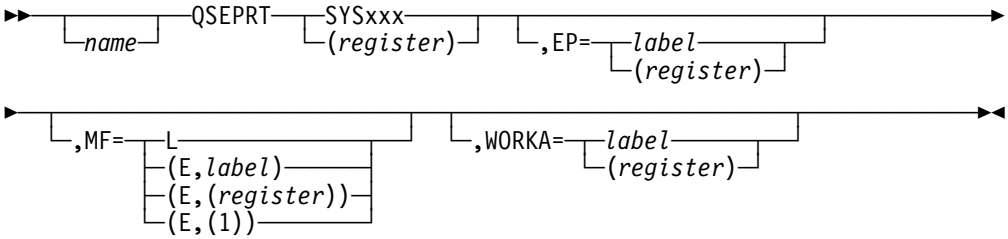
QSETPRT (Query Printer Setup) Macro

The purpose of the QSETPRT macro is to determine how the printer is set up and/or to build a SETPRT parameter list that can later be passed to the SETPRT routine to recreate the current setup. The parameter list passed to QSETPRT is converted to a SETPRT parameter list with INIT=Y set in it. FCB verification and SEP indicators are not set in the parameter list.

The SETPRT and QSETPRT macros may use registers 0, 1, 14, and 15 in their expansions. The caller's addressability must be based on a different register.

After successful completion of QSETPRT, register one points to the parameter list passed to QSETPRT. This means that after issuance of the standard form of QSETPRT, the parameter list can later be used with the execute form of SETPRT. Even though successful issuance of QSETPRT converts the parameter list to SETPRT, a subsequent reexecution of the same QSETPRT macro will still act as QSETPRT.

The user may specify either the LUB ID or the LUB address when calling QSETPRT. However, on successful completion, the parameter list always contains the LUB ID (in the SPPLUNIT field).



The parameters have the same meaning as for the SETPRT macro except that MF may refer to a parameter list created by coding MF=L on either a SETPRT or QSETPRT macro. Successful execution of the QSETPRT routine effectively changes the parameter list to a SETPRT parameter list. It can then be used either for SETPRT or another QSETPRT. A QSETPRT parameter list that has not been passed to QSETPRT with a return code of zero cannot be used as a SETPRT parameter list but for another QSETPRT.

The possible QSETPRT return codes are 0, 8, C, 14, 28, 2C, 30, 34, and 38. See the SETPRT return codes.

Return Codes

When SETPRT or QSETPRT returns to its caller, register 15 contains a return code as described below. With current support, the bytes described with ".." contain zeros but to allow for possible future support, application programs should not assume those bytes to contain zeros.

All return codes except 0, 30, 38 and 0028 have associated messages that are written on the printer and on SYSLOG.

.

Return Code	Meaning
00000000	Successful completion due to one of the following: • Output device is 3800 and all functions completed successfully. • Output device is PRT1 printer. If the FCB parameter was specified, and it was not a request for the system default forms control buffer phase, then the LFCB macro was issued successfully. • Caller is executing in the B-transient area and no operation was performed.
......04	All printer setup requests were successfully completed except for the burster request. The burster request was not done because there is no Burster- Trimmer-Stacker installed on the 3800.
......08	Invalid device type. The output device must be 3800 or PRT1 printer.
......0C	Invalid parameter list. The length value in the list is not valid, the logical unit was not specified, the list is not a word boundary, or reserved field does not contain zeros.
ggcctt10	Phase not found in the library or the phase header has an invalid format. The header is the first eight bytes. Byte 2 of register 15 indicates the type of phase that could not be found or has an invalid format:

Byte 2 (tt)	Phase Type
04	Forms control buffer (FCB) phase.
08	Copy modification phase.
0C	Character arrangement table phase.
10	Graphic character modification phase.

If the phase type code is 0C or 10, then bytes 0 and 1 (ggcc) identify which of the possible character arrangement table (CAT) phases or graphic character modification (GCM) phases was required. If a CAT phase was required, byte 0 (gg) is zero and byte 1 (cc) identifies the character arrangement table phase (that is, 01 for the first CAT, etc.). If the CAT that was required was specified in the MODIFY keyword and not the CHARS keyword, (cc) is set to 05.

If a graphic character modification phase was required, then byte 1 (cc) identifies the CAT for which the GCM phase was being loaded from the library and byte 0 (gg) identifies which of the four possible GCM phases was required.

ggccop14	Permanent I/O error on printer. Byte 2 (op) of register 15 contains the channel command code of the failing CCW. For example, if the printer gives an error on a Load Copy Modification channel command, then byte 2 (op) contains X'35'. If byte 2 (op) is X'83' or X'25', then bytes 0 and 1 have the same meaning as for a 10 return code.

.....18	The operator canceled the SETPRT request because the manual setup could not be performed.
.....1C	Reserved. Should not occur.
..ccnn20	More character generation storage was requested than was available on the printer. The cc identifies the character arrangement table that caused the error (that is, 01 for the first CAT, etc.). If the table is the one specified in the MODIFY keyword and not the CHARS keyword, cc is 05. The nn is either 2 or 4 and indicates the number of WCGMs available on the device.
..cc..24	A byte in a character arrangement table references a character generation module (CGM) that was not identified in the table. This should never occur for character arrangement tables created by the IEBIMAGE utility. The cc identifies the character arrangement table that caused the error (that is, 01 for the first CAT, etc.). If the table is the one specified in the MODIFY keyword and not the CHARS keyword, cc is 05.
....ss28	Not enough storage was available to perform printer setup. The ss is 00 if the initial 512-byte work area could not be obtained, or 04 if the secondary 11776-byte area could not be obtained. Decrease the value of SIZE on the EXEC statement, run the program in a larger partition, or supply a valid work area to SETPRT with the WORKA parameter.
....uu2C	Symbolic unit is invalid or not assigned. The uu is 04 if the symbolic unit is invalid or 08 if the symbolic unit is not assigned.
....cc30	SETPRT or QSETPRT routine is not in the System Virtual AREA and could not be loaded from the private or system library. Byte 2 is the nonzero return code from the CDLOAD macro.
..yyzz34	Internal macro failure. This should never occur. yy is the internal macro's return code. zz indicates failing macro where 04 is the EXTRACT macro, 08 is the MODCTB macro, and 0C is the CDLOAD macro. For information on EXTRACT or MODCTB return codes, see *VSE/ESA Messages and Codes*
.....38	User-supplied work area is not a double-word boundary.
....rr3C	PRT1 initialization failed. SETPRT issued an LFCB macro because the output device is a PRT1 printer and the FCB parameter was specified. The LFCB routine gave nonzero return code rr. For an explanation of the LFCB return codes, see “Return Codes in Register 15” on page 251.

Calling SETPRT for a VSE/POWER-Controlled Printer

Some, but not all possible errors are detectable by SETPRT when the 3800 is controlled by VSE/POWER. Some user errors are detectable only when VSE/POWER attempts to write the file on the real printer. Such errors include:

- Non-existent control phase
- Control phase contains invalid data

- Unavailable forms or forms overlay frame
- Burster-Trimmer-Stacker not installed

Control phases from private libraries should not be requested unless the library is allocated to the VSE/POWER partition.

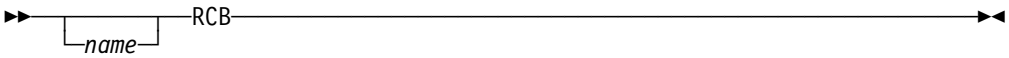
A work area of 512 bytes is always enough, whether supplied via the WORKA parameter or gotten via GETVIS by SETPRT routine.

When the device status is changed for any of the following, the output is segmented:

- BURST
- FORMS
- FLASH
- Copy grouping
- CINDX

This means that the data already sent is queued for output and a new file is begun with the same job attributes. The new SETPRT parameter list is written as the first record of the new file. When a CINDX value that is greater than 1 is passed to VSE/POWER, the output is segmented as described, but the VSE/POWER copy count is set to 1.

RCB (Resource Control Block Definition) Macro



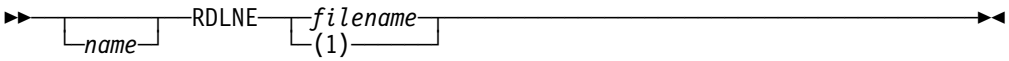
Required RMODE: **24** or **ANY**

The RCB macro generates an 8-byte word-aligned resource control block (RCB) which allows you to protect a user-defined resource if the ENQ macro is issued before (and the DEQ macro is issued after) each use of the resource. The RCB may be allocated below or above the 16MB line.

The format of the RCB is shown below.

Bytes	Purpose of Bits
0	All bits are set to 1 to indicate that the resource has been placed in a priority queue by the ENQ macro.
1-3	Reserved.
4-7	ECB address of current resource owner (4-byte address). Bit 0 of byte 4 will be set to 1 when another task is waiting to use the resource.

RDLNE (Read a Line) Macro



Requirements for the caller:

AMODE: 24

RMODE: 24

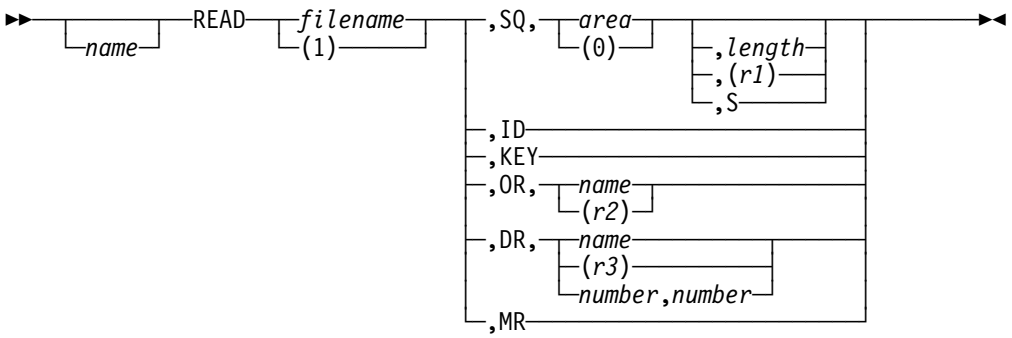
ASC Mode: Primary

The RDLNE macro provides selective online correction when a journal tape is being processed on the IBM 1287 Optical Reader. This macro reads a line in the online correction mode while processing in the offline correction mode. RDLNE should be used in the COREXIT routine only, or else the line following the one in error will be read in online correction mode.

If your IBM 1287 cannot read a character, IOCS first resets the input area to binary zeros and then retries the line containing the character that could not be read. If the read is unsuccessful, you are informed of this condition via your error correction routine (specified in DTFOR COREXIT). The RDLNE macro may then be issued to cause another attempt to read the line. If the character in the line still cannot be read, the character is displayed on the 1287 display scope. The operator keys in the correct character, if possible. If the operator cannot readily identify the defective character, the reject character may be entered in the error line. This condition is posted in filename+80 for your examination. Wrong-length records and incomplete read conditions are also posted in filename+80.

filename | (1)
The symbolic name of your 1287 file from which the record is to be retrieved. This name is the same as that specified in the DTFOR header entry for the file.

READ (Read a Record) Macro



Requirements for the caller:

AMODE: 24

RMODE: 24

ASC Mode: Primary

The READ macro transfers a record or part of a record from an input file to an area in virtual storage. Do not code a READ macro after a WRITE SQ macro.

filename | (1)
Specifies the name of the file from which the record is to be read. The name is the same as the one you specified in the DTFxx macro for the file.

SQ Required for sequential work files.

area | (0)
The name of the input area used by a sequential file.

length | (r1)| S
Species only sequential files of undefined format (RECFORM=UNDEF). Specifies the actual number of bytes to be read, or the register where the number is to be found. S specifies that the entire record is to be read. The length of the record is taken from the filenameL field of the DTF.

ID Applies to DAM. Specifies that the reference is to be by ID (identifier in the count area of the record).

KEY For ISAM, KEY is required. For DAM, KEY specifies that the record reference is to be by record key (control information in the key area of the DASD record).

OR Indicates that the file to be accessed is on an IBM 1287 or 1288 optical character reader.

name | (r2)
Specifies the CCW list address to be used to read a document from a file on your IBM 1287 or 1288.

DR,name | (r3) | number,number
Indicates an IBM 3886 Optical Character Reader is the input device.

The line number to be read and the format record for the line are specified in one of three ways:

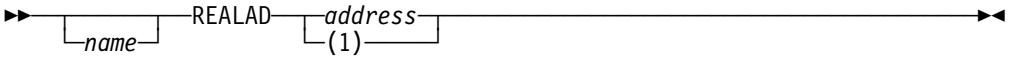
name Provides the symbolic address of a two-byte hexadecimal field containing the line number in the first byte and the format record number in the second byte.

(r3) Provides the number of the register that contains the address of the two-byte hexadecimal field.

number,number
Provides the decimal line number to be read (any number from 1 through 33), followed by the format record number used to read the line (0-63).

MR Specifies that the file is for a magnetic ink character reader (MICR).

REALAD (Real Address Return) Macro



Requirements for the caller:

AMODE: 24 or 31

RMODE: 24 or ANY

ASC Mode: Primary

The macro returns the real address corresponding to a specified virtual address.

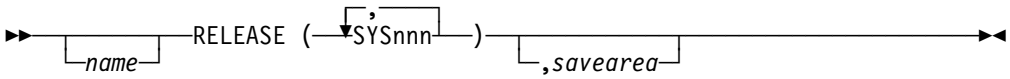
address | (1)
Is the virtual address to be converted. It can be specified as a symbol or in register notation.

Register 0 returns the 31-bit real address corresponding to the specified virtual address if and only if the virtual address points to a PFIXed page, otherwise register 0 contains 0. Thus, the macro can be used to test if a page is PFIXed.

The AMODE/RMODE of the caller can be 24 or 31 bit. When called in 24-bit mode, the address will be treated as a 3-byte address; when called in 31-bit mode, the address will be treated as a 4-byte address.

Note: The pages of a partition running in real mode are treated as if they were fixed.

RELEASE (Release Logical Unit) Macro



Requirements for the caller:

AMODE: 24

RMODE: 24

ASC Mode: Primary

The macro specifies the names of programmer logical units to be released. RELEASE may be used only for units used within a given partition.

Consider informing your operator about this release of an assignment. You can do this by a message to the console.

SYSnnn

Specifies the programmer logical unit that is to be released. Up to 16 units may be specified in a list, which must be enclosed in parentheses.

All the units specified are checked by the assembler to assure that no system logical units are requested for release. If system logical units are specified, an MNOTE is issued and such units are ignored. Before any release is attempted, a check is made for ownership of the unit. If the requesting partition does not own the unit, or if the unit is already unassigned, the request is ignored.

savearea

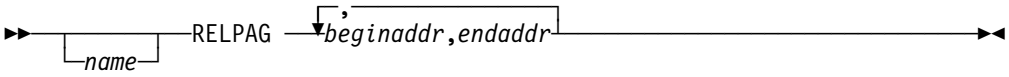
Is the name of an 8-byte word-aligned area where registers 0 and 1 are saved for your program. If the operand is not provided, the contents of registers 0 and 1 are overwritten.

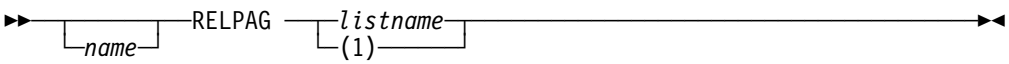
The macro expansion includes a unit table and loads the table's address into register 0. If the save area operand is specified, the macro expansion saves registers 0 and 1.

If there is no permanent assignment, the device is unassigned. If the device is at permanent assignment level, no action is taken on the unit.

RELPAG (Release Page) Macro

You can code the macro in either of the following formats:





Requirements for the caller:

AMODE: 24 (if SPLEVEL SET=1)

24 or 31 (if SPLEVEL SET>1)

RMODE: 24 (if SPLEVEL SET=1)

24 or ANY (if SPLEVEL SET>1)

ASC Mode: Primary

The RELPAG macro causes the contents of one or more virtual storage areas to be released. If the affected areas are in real storage when the RELPAG macro is executed, their contents are not saved but are overwritten when the associated page frames are needed to satisfy pending page frame requests.

After the RELPAG macro has been executed for an area and a location in that area is referenced again during the current program execution, the related page is attached to a page frame which contains all zeros.

The storage area is released only if it contains at least one full page. You can be sure of this only if the specified area is two times the page size minus 1 or bigger. This is explained in more detail under “FCEPGOUT (Force Page Out) Macro” on page 191.

beginaddr
Points to the first byte of the area to be released.

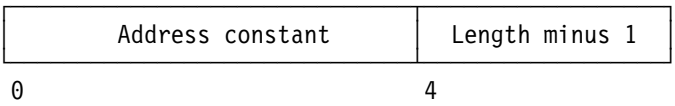
endaddr
Points to the last byte of the area to be released.

listname | (1)
Is the name of a list of consecutive 8-byte entries as shown below. (Register notation may also be used.) The address of this parameter list and the addresses in the list are treated as 3-byte addresses if the macro is invoked in 24-bit addressing mode and as 4-byte addresses if invoked in 31-bit addressing mode.

24-bit Addressing Mode:

X'00'	Address constant	Length minus 1
0	1	4

31-bit Addressing Mode:



where:

Address constant =
Address of the first byte of the area to be released.

Length = A binary constant indicating the length of the area to be released.

The end of the list is indicated by a non-zero byte following the last entry (for 24-bit addressing mode). For 31-bit addressing mode, a non-zero value in bit 0 of the byte following the last entry indicates the end of the list.

Exceptional Conditions

- The program is running in real mode.
- The area is, fully or partially, outside of the virtual partition of the requesting program.
- A page handling request is pending for the referenced page(s).
- The page(s) is (are) fixed. For these pages, the RELPAG request will be ignored.

Return Codes in Register 15

0 All referenced pages have been released or the request has been ignored because the requesting program is running in real mode.

2 The begin address is greater than the end address, or a negative length has been found.

4 The area, fully or partially, does not belong to the partition where the issuing program is running. The release request has only been executed for those pages which belong to the partition of the issuing program.

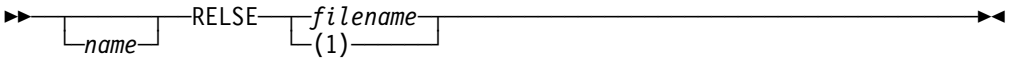
8 At least one of the requested pages is temporarily fixed, either by the system or by a previous PFIX macro. The release request has only been executed for the unfixed pages.

16 List of areas that are to be released is not completely in the requesting program's partition. The request is ignored.

20 Inconsistent function/option code provided in register 15; the request is ignored (this is not possible if the macro interface is used).

Any combination of the return codes 2, 4, and 8 is possible.

RELSE (Release a Block) Macro



Requirements for the caller:

AMODE: 24

RMODE: 24

ASC Mode: Primary

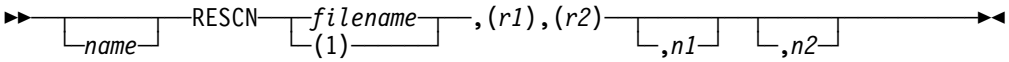
The macro is used with blocked input records read from or updated on a disk device. The macro is also used with blocked input records read from magnetic tape.

The macro allows you to skip the remaining records in a block and continue processing with the first record of the next block when the next GET macro is issued. When used with blocked spanned records, RELSE causes the next GET to skip to the first segment of the next record.

If RELSE immediately precedes a CNTRL macro with FSL or BSL (tape spacing for spanned records), then LIOCS ignores the CNTRL macro request.

filename | (1)
Specifies the name of the file for which a release operation is requested. The name is the same as the one you specified in the DTFxx macro for the file. It is the only operand required for this macro and can be specified as a symbol or in register notation.

RESCN (Re-Scan) Macro



Requirements for the caller:

AMODE: 24

RMODE: 24

ASC Mode: Primary

The macro selectively rereads a field on a document if one or more defective characters make this type of operation necessary. The field is always right-justified into the area (normally within IOAREA1) that was originally intended for this field as specified in the CCW. The macro first resets this area to binary zeros.

Note: If you use the macro with a READ FORWARD command, the input area is not cleared.

filename | (1)
Specifies the name of the document file for which a reread operation is requested. The name is the same as the one you specified in the DTFOR macro for the file; it can be specified as a symbol or in register notation.

(r1) Specifies a general-purpose register from 2 to 12 into which the program places the address of the load format CCW.

(r2) Specifies a general-purpose register from 2 to 12 into which the program places the address of the load format CCW for reading the reference mark.

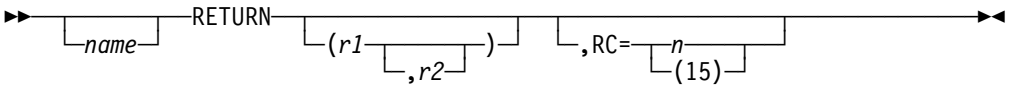
n1 Allows you to specify the number of attempts (one to nine allowed) to reread the unreadable field. If this operand is omitted, one is assumed. If n1 is omitted but n2 is specified, a comma must be coded instead of n1 to indicate its absence.

n2 This operand cannot be used for a file on your IBM 1288.

The operand indicates one more reread which forces online correction of any unreadable character(s) by individually projecting the unreadable character(s) on the display scope of your IBM 1287.

The operator must key in a correction (or reject) character(s).

RETURN (Return after Call) Macro



Requirements for the caller:

AMODE: 24 or 31

RMODE: 24 or ANY

ASC Mode: Primary

The RETURN macro restores the registers whose contents were saved and returns control to the calling program.

r1|(r1,r2)
Specify the range of the registers to be restored from the save area of the program that receives control. The operands are written as self-defining values. The operands cause the desired registers in the range from 14 through 12 (14, 15, 0 through 12) to be restored from words 4 through 18 of the save area.

If r2 is omitted, only the register specified by r1 is restored. If you omit the whole operand, the contents of the registers are not changed.

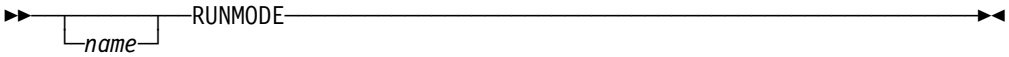
To access this save area, the system requires that register 13 contains the save area address. Therefore, in your program, load the address of the save area into register 13 before you issue the RETURN macro.

RC=n | (15)
Indicates a user-specified return code to be passed to the calling program. **n** must be a number between 0 and 4095, which is placed right-adjusted in register 15.

If register notation is used, the register is loaded into register 15.

If RC is specified, register 15 is not restored from the save area.

RUNMODE (Run-Mode Indication) Macro



Requirements for the caller:

AMODE: 24

RMODE: 24

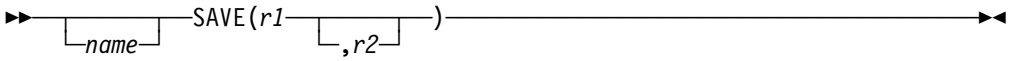
ASC Mode: Primary

The macro returns the following information to the program issuing this macro:

- Register 1 contains 0 if the issuing program is running in virtual mode.
- Register 1 contains 4 if the issuing program is running in real mode.

No operand is required for this macro.

SAVE (Save Register) Macro



Requirements for the caller:

AMODE: 24 or 31

RMODE: 24 or ANY

ASC Mode: Primary

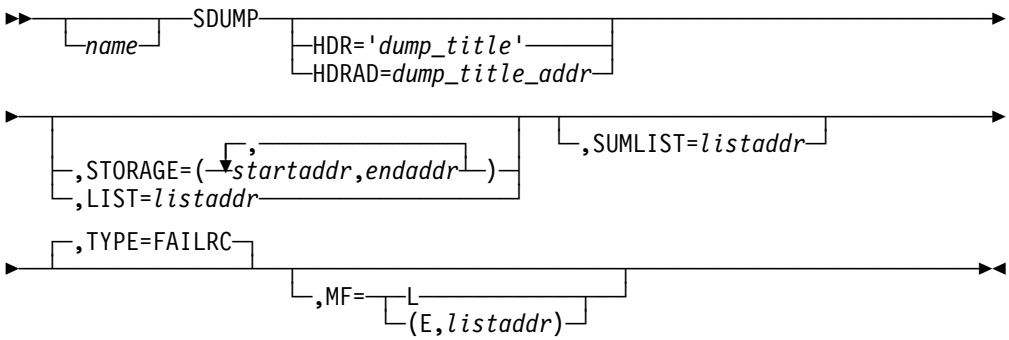
The macro stores the contents of specified registers in the save area provided by the calling program.

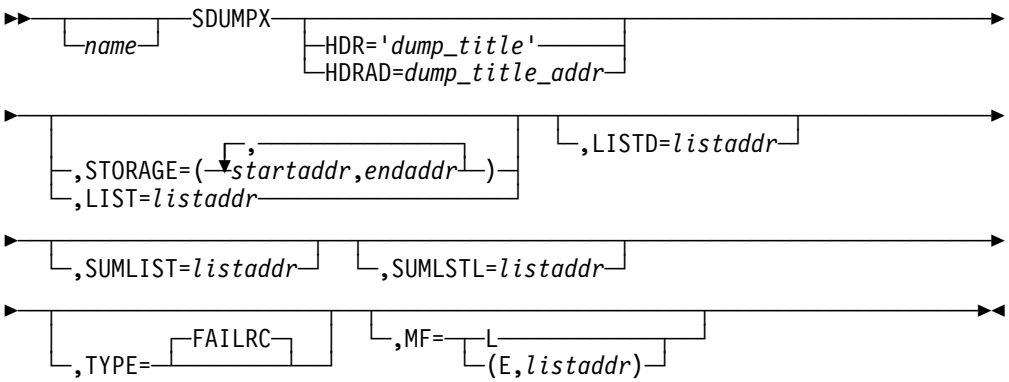
The operands r1,r2 specify the range of the registers to be stored in the save area of the calling program. The address of this area is passed to the program in register 13. The operands are written as self-defining values so that they cause desired registers in the range of 14 through 12 (14, 15, 0 through 12) to be stored when inserted in an STM assembler instruction.

Registers 14 and 15, if specified, are saved in words 4 and 5 of the save area. Registers 0 through 12 are saved in words 6 through 18 of the save area. The contents of a given register are always stored in a particular word in the save area. For example, register 3 is always saved in word 9 even if register 2 is not saved.

If r2 is omitted, only the register specified by r1 is saved.

SDUMP/SDUMPX





Requirements for the caller:

AMODE: 24 or 31

RMODE: 24 or ANY

ASC Mode: Primary or AR (access register)

The SDUMP and SDUMPX macros provide a dump of virtual storage which contains user data and system data. The macros provide support for 31-bit addressing. The dump output is written either into a dump library or onto SYSLST, depending on the current setting of the SYSDUMP/SYSDUMPC option. The High Level Assembler is required to compile the macros.

The SDUMP macro causes address ranges in your current address space to be dumped, whereas the SDUMPX macro also provides for storage ranges in address spaces or data spaces to which addressability via an ALET or via an STOKEN exists.

If the program is running in primary ASC (address space control) mode, either SDUMP or SDUMPX can be used. Otherwise, when the program runs in access register (AR) mode, the SDUMPX macro must be used. SDUMPX provides all of the functions of SDUMP, but generates code and addresses that are appropriate for AR mode.

If you are in access register (AR) mode, issue the SYSSTATE ASCENV=AR macro before you issue the SDUMPX macro to tell SDUMPX to generate code appropriate for AR mode.

The SDUMP macro cannot dump data space storage. To dump data space storage, SDUMPX is to be used instead by including either the LISTD or SUMLSTL operand.

When SDUMPX is entered, the specified parameter list and all areas to which the list points must be in the current address space of your partition.

HDR='dump-title'

Specifies the title or header to be used for the dump. The title must be from 1 to 100 characters, enclosed in quotes.

Note: The header is displayed only if the dump is routed to SYSLST (OPTION NOSYSDMP). If the dump is routed to a dump sublibrary,

the header is not displayed; you can, however, inspect it by using the SELECT DUMP SYMPTOMS function of Info/Analysis.

HDRAD=dump-title-addr

Specifies the address of the title to be used for the dump. The dump title field consists of a one-byte length field, followed by the specified title. The length field specifies the length of the title, excluding the length byte itself. The macro accepts a length value of 1 to 100. See Note under HDR.

STORAGE=(startaddr,endaddr,...)

Specifies the start and end address of the virtual storage area to be dumped. One or more pairs of start and end addresses may be given, that is, the list must contain an even number of addresses, and each address must occupy one fullword. For example:

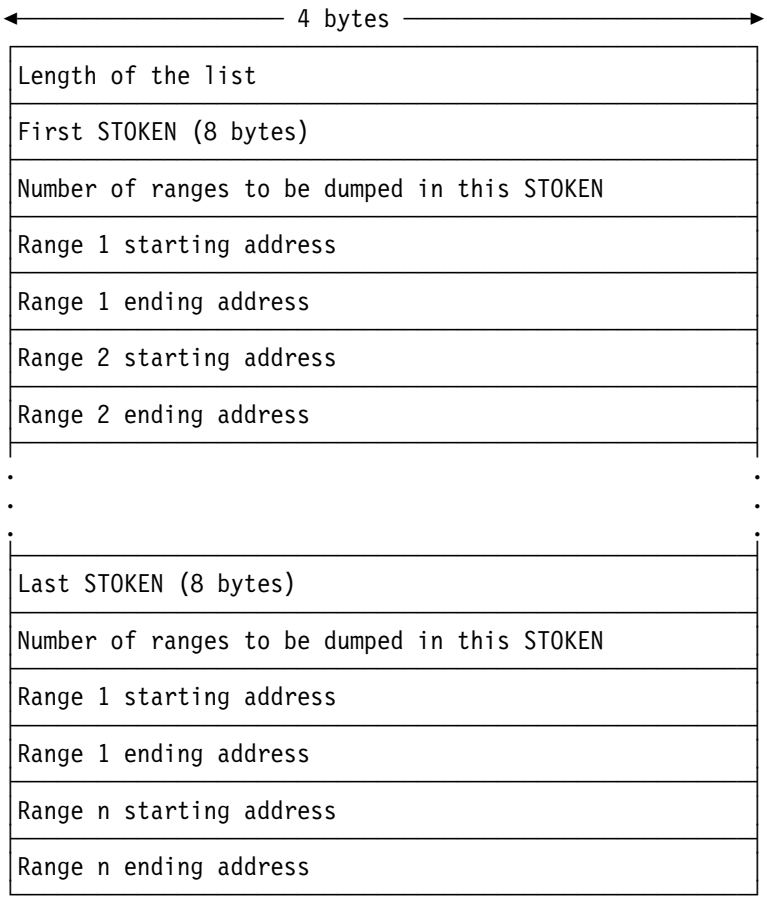
```
STORAGE=(startaddr,endaddr,startaddr,endaddr)
```

LIST=listaddr

Specifies a list of start and end addresses of the storage to be dumped. The list must contain an even number of addresses, and each address must occupy one fullword. In the list, the high-order bit of the fullword containing the last ending address of the list must be set to 1; all other high-order bits must be set to 0.

LISTD=listaddr

Specifies a list of start and end addresses - qualified by STOKENs (space tokens) - of the areas to be included in the dump. Specify the STOKENs and address ranges as follows:



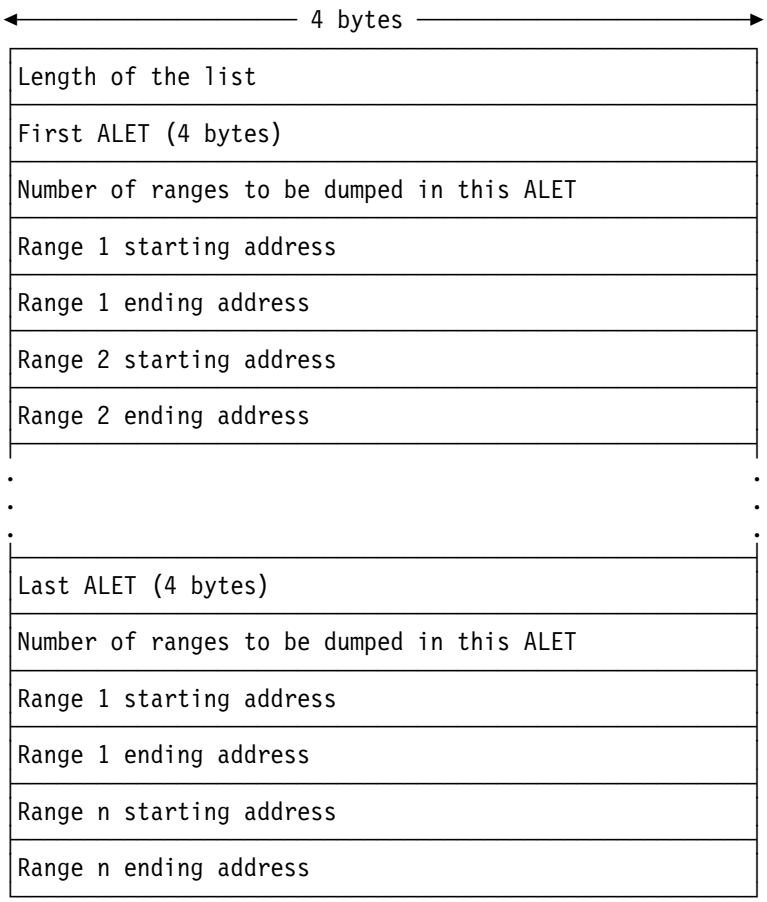
The first fullword of the list contains the number of bytes (including the first word) in the list. STOKEN refers to any address or data space.

SUMLIST=listaddr

Specifies a list of start and end addresses of areas to be included in the dump. The list must contain an even number of addresses, and each address must occupy one fullword. In the list, the high-order bit of the fullword containing the last ending address of the list must be set to 1; all other high-order bits must be set to 0.

SUMLSTL=listaddr

Specifies a list of start and end addresses - qualified by ALETs (access list entry tokens) - of areas to be included in the dump. Specify the ALETs and address ranges as follows:



The first fullword of the list contains the number of bytes (including the first word) in the list. ALET refers to entries in either a DU-AL or a PASN-AL associated with any address or data space that the caller has addressability to.

TYPE=FAILRC

Specifies that the macro is to return a reason code together with the return code in register 15. The reason code explains why the dump failed. This is also the default if the TYPE operand is omitted.

MF=L | (E,listaddr)

L specifies the **list** form of the macro, which generates a control program parameter list that can be used by the execute form of the macro. In the list

form, only A-type addresses may be used; RX-type addresses and registers cannot be used.

E specifies the **execute** form of the macro, which uses the parameter list generated by the list form of the macro. **listaddr** specifies the address of the parameter list.

If the MF operand is omitted, the standard form of the macro is used, which is for programs that are not reenterable or that do not change values in the parameter list.

Return Codes in Register 15

Register 15 contains one of the following return codes in bits 24-31:

0 A complete dump was taken.
4 A partial dump was taken.
8 The system was unable to take a dump.

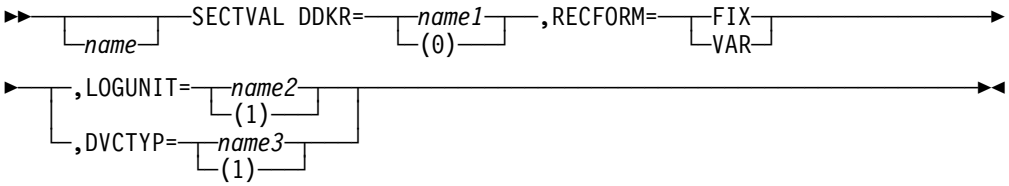
For return code 8, a reason code is supplied in bits 16-23 of register 15. The reason code explains why the dump failed.

The reason codes are as follows:

Figure 48. SDUMP Reason Codes for Return Code 8

Reason Code	Meaning
15	The parameter list address is zero.
16	Invalid parameter list options - invalid dump type.
18	Invalid address range specified: conflicting begin and end addresses.
19	Invalid user data specified - header is not correct.
1B	Invalid STORAGE list or LISTD.
1E	Invalid address space or data space.
2A	The caller-supplied storage list is inaccessible.
2B	The user header data is inaccessible.
2E	The caller-supplied SUMLIST is inaccessible.
33	Out of space limits.
34	The caller-supplied STOKEN and range list in LISTD is inaccessible.
35	The caller-supplied ALET and range list in SUMLSTL is inaccessible.
E0	SYSLST is not available.
E1	Dump library is full - part of dump is on SYSLST.
E2	Dump library is not defined - dump is on SYSLST.
E3	Dump library is in error - dump is on SYSLST.
E4	Getvis Failure
E5	No dump due to JCL/Macro options

SECTVAL (Sector-Value Calculation) Macro



Requirements for the caller:

AMODE: 24 or 31

RMODE: 24 or ANY

ASC Mode: Primary

The macro calculates the sector value of the start of the requested record on the track of a disk storage device when RPS is used. The macro returns this value in register 0. If an error is detected in calculating the sector value, the system returns a no-operation sector value (X'FF').

The sector value is calculated from data length, key length, and record number (except record number 0) information. Values are calculated for fixed or variable-length and for keyed and non-keyed records.

DDKR=name1 | (0)

The information needed to calculate the sector value should be specified in the 4-byte field at name1, or in the specified register. The four bytes of information have the format ddkr, where

dd = Two-byte field which specifies for:

- **Fixed-length** records, the data length of each record.
- **Variable-length** records, the total number of **user** data bytes written on the track after R0 up to the requested record.

If the k and r bytes are zero, the dd field is considered a 16-bit number specifying the total number of bytes (including all gaps, keys, and overhead) that were written on the track after R0 and the home address. Therefore, a sector value should **not** be requested for record 0 (fields kr=0000).

k = A one-byte field indicating the key length for:

- **Fixed-length** records, the actual key length must be specified.
- **Variable-length** records, any non-zero value is sufficient to indicate the presence of keys.

Note: For non-keyed records the value should be 0.

r = A one-byte record number field which specifies the number of the record of which the sector value is being requested. Do not specify record number 0, unless you use the special interface described above.

RECFORM=FIX | VAR
Specifies whether the records are of fixed- or variable length format.

LOGUNIT=name2 | (1)
Specifies the logical unit number of the device, either in a two-byte field at **name2** or in the two low-order bytes of the specified register (that is, right-adjusted). See Note.

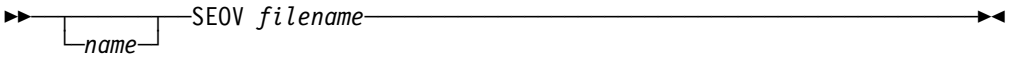
DVCTYP=name3 | (1)
Specifies the DTF device type code, either in a one-byte field at **name3** or in the low-order byte of the specified register (that is, right-adjusted). The remaining bytes of that register are assumed to be zero. See Note.

The following device type codes (the same as those contained in the DTF block) are valid for IBM disk devices as indicated:

IBM Device	Code
3375	X'0B'
3380	X'0C'
3390	Use the LOGUNIT operand or the GETVCE macro
3390 Model 1	X'26'
3390 Model 2	X'27'
3390 Model 3	X'24'
3390 Model 9	X'25'
9345	X'04'

Note: If LOGUNIT or DVCTYP are not supplied in a register, the address of the named field must not be based on register 1.

SEOV (System End-of-Volume) Macro



Requirements for the caller:

AMODE: 24

RMODE: 24

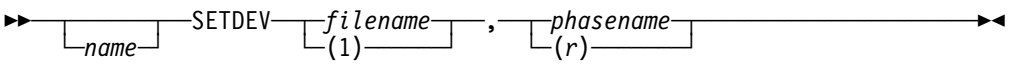
ASC Mode: Primary

The macro may be used only with physical IOCS to automatically switch volumes if SYSLST or SYSPCH are assigned to a tape output file. SEOV writes a tapemark, rewinds and unloads the tape, and checks for an alternate tape. If none is found, a message is issued to the operator who can mount a new tape on the same drive and continue.

If an alternate unit is assigned, the macro fetches the alternate switching routine to promote the alternate unit, opens the new tape, and makes it ready for processing. When using this macro, you must check for the end-of-volume condition in the CCB.

filename | (1)
Specifies the name of the file for which the end-of-volume operation is requested. The name is the same as that specified as name in the DTFPH macro for the file.

SETDEV (Set Device) Macro



Requirements for the caller:

AMODE: 24

RMODE: 24

ASC Mode: Primary

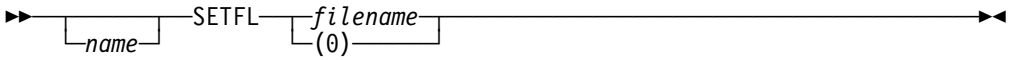
The macro changes format records during execution of the program. When the new format record has been loaded into your IBM 3886, control returns to the next sequential instruction in your program. If the operation is not successful, the completion code is posted at EXITIND and control is passed to the COREXIT routine, or the job is canceled. If you issue the SETDEV macro and no documents remain to be processed and the end-of-file key has been pressed on the device, control is passed to the end-of-file routine.

filename | (1)
Specify the name you used for the DTFDR macro which defines the file. Register notation must be used if your program is to be self-relocating.

phasename | (r)
The operand either:

- Specifies the name of the format record to be loaded, or
- Indicates the register containing the address of an eight-byte area that contains this name.

SETFL (Set File Load Mode) Macro



Requirements for the caller:

AMODE: 24

RMODE: 24

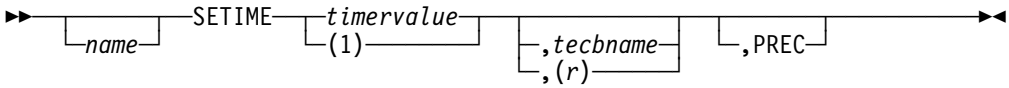
ASC Mode: Primary

The macro causes ISAM to set up the file so that the load or extension function can be performed. This macro must be issued whenever the file is loaded or extended.

When loading a file, SETFL pre-formats the last track of each track index. When extending a file, SETFL preformats only the last track of the last track index plus each new track index for the extension of the file. This allows prime data on a shared track to be referenced even though no track indexes exist on the shared track.

filename | (0)
The name of the file to be loaded is the only operand required for this macro. The name you specify is the same you used for the DTFIS macro which defines the file. This name can be specified as a symbol or in register notation. Register notation is necessary if your program is to be self-relocating.

SETIME (Set Interval Timer) Macro



Requirements for the caller:

AMODE: 24 or 31

RMODE: 24 or ANY

ASC Mode: Primary

The SETIME macro sets the interval timer to the specified value. If the tecbname operand is specified, bit 0 of byte 2 in the TECB is set to 0 so that a subsequent WAIT/WAITM macro can be issued by the task issuing the SETIME macro. When the interval specified in timervalue elapses, the interrupt routine is entered or the TECB is posted. If tecbname is omitted, the interrupt routine specified in a previous

STXIT IT macro is entered (if no STXIT IT macro was issued prior to the time of the interrupt, the interrupt is ignored). If tecbname is specified, the TECB is posted (byte 2, bit 0 of the TECB is set to 1) and the task is posted ready to run if it is already waiting.

Warning

The SETIME macro should not be issued from an application running under CICS. Issuing this macro can adversely affect CICS execution.

timervalue | (1)
Specifies the amount of time for the interval. This value can be specified either as an absolute expression or in register notation. If register notation is used, the pertinent register must contain the time value.

The largest allowable value is 55 924 seconds (equivalent to 15 hours, 32 minutes, 4 seconds) if PREC is omitted, and 8 388 607 (equivalent to 7 hours, 46 minutes, 2 seconds) if PREC is specified.

tecbname | (r)
Specifies the name (address if register notation is used) of a timer event control block (TECB) which must have been defined previously in your program by a TECB macro. If you use register notation, register 0 and 1 must not be used. After having executed the SETIME macro, the system returns the TECB address in register 1.

The address of the tecbname is treated as a 3-byte address if SETIME is issued in 24-bit mode, and as a 4-byte address if SETIME is issued in 31-bit mode.

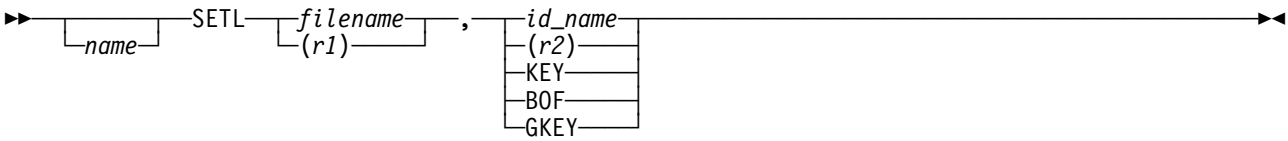
A SETIME macro **without** the tecbname operand is used in combination with a previous STXIT IT macro.

Any previous STXIT IT specification is overwritten when using the SETIME macro **with** the tecbname operand.

If you omit tecbname but want to specify PREC, you must code a comma instead of tecbname to indicate the omission.

PREC Indicates that the timer value specified in the first operand is expressed in 1/300 of a second. When PREC is omitted, the timer value is in seconds.

SETL (Set Limits) Macro



Requirements for the caller:

AMODE: 24

RMODE: 24

ASC Mode: Primary

The macro initiates the mode for sequential retrieval and initializes ISAM to begin retrieval at the specified starting address.

Note: Sequential processing must always be terminated by issuing an ESETL macro. The ESETL (end set limit) macro should be issued before issuing a READ or WRITE if ADDRTR and/or RANSEQ are specified in the same DTF. Another SETL can be issued to restart sequential retrieval.

filename | (r1)
Specifies the same name as that used in the DTFIS header entry, as a symbol or in register notation. Register notation is necessary if your program is to be self-relocating.

id-name | (r2)
Specifies that processing is by record ID. The operand specifies the symbolic name of the eight-byte field in which you supply the starting (or lowest) reference for ISAM use. This field contains the information shown in Figure 49 on page 368.

KEY Specifies that processing begins with a key you supply. The key is supplied in the field specified by the DTFIS KEYARG operand. If the specified key is not present in the file, an indication is given at filenameC.

BOF Specifies that retrieval is to start at the beginning of the logical file.

GKEY Indicates that selected groups of records within a file containing identical characters or data in the first locations of each key can be selected. GKEY allows processing to begin at the first record (or key) within the desired group.

You must supply a key that identifies the significant (high order) bytes of the required group of keys. The remainder (or insignificant) bytes of the key must be padded with blanks, binary zeros, or bytes lower in collating sequence than any of the insignificant bytes in the first key of the group to be processed. For example, a GKEY specification of D6420000 would permit processing to begin at the first record (or key) containing D642xxxx, regardless of the characters represented by the x's.

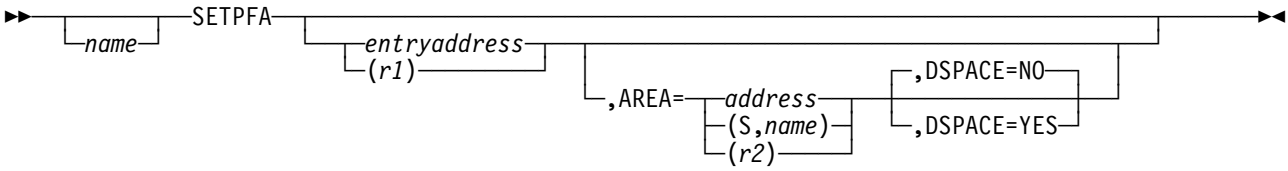
Your program must determine when the generic group is completed. Otherwise, ISAM continues through the remainder of the file.

Note: If the search key is greater than the highest key on the file, the filename status byte is set to X'10' (no record found).

Byte	Iden-tifier	Contents in Hexadecimal	Information
0	m	02-F5	Number of the extent in which the starting record is located.
1-2		0000 (disk)	Always zero for disk
3-4	cc	Cylinder number for disk:	
		0000-00C7	For IBM 2311, 2314/2319: 0-199
		0000-0193	For IBM 3330/3333: 0-403
		0000-015B	For IBM 3340 with 3348 Model 35: 0-347
		0000-02B7	For IBM 3340 with 3348 Model 70: 0-695
5-6	hh	Head position for disk:	
		0000-0009	For IBM 2311.
		0000-0013	For IBM 2314/2319.
		0000-0012	For IBM 3330/3333.
		0000-000B	For IBM 3340.
7	r	01-FF	Record location

Figure 49. Field Supplied for SETL Processing by Record ID

SETPFA (Set Link to Page-Fault Appendage) Macro



Requirements for the caller:

AMODE: 24

RMODE: 24

ASC Mode: Primary

The macro either sets up or removes linkage to a user-written page-fault appendage routine. When this linkage is set up, your page-fault appendage routine receives control each time a page fault occurs in the task that owns the routine. It receives control also when the system has completed handling such a page fault.

For a page fault while the supervisor is servicing the owning task, the appendage routine does not receive control. The same may apply to an IBM-supplied base program such as VTAM*.

As long as the exit is active, your program does not lose control when a page fault occurs. Instead, the system:

1. Returns control to your program immediately.
2. Passes to your routine the page fault information which otherwise serves as input to the page management routine of VSE.

A page-fault appendage routine may be useful in a performance critical application. The routine can set the conditions for the dispatching of a private (not VSE controlled) subtask.

entryaddress | (r1)

By specifying an entry address, you activate the page-fault exit for your program. The macro causes the system to set up the required linkage to the page-fault appendage routine of your program at the specified address.

If SETPFA is issued without an operand, the linkage to the page fault appendage is terminated. Each issuance of SETPFA supersedes all previous SETPFAs for that task. Only one task per partition is allowed to have a page-fault appendage.

Restriction: The entry address must be a 24-bit address.

The entry address may be specified as a symbol or in register notation.

AREA=address | (S,name) | (r2)

Specifies the address of a 16-byte area that contains (in bytes 0-3) the virtual address of the page to be handled, and (in bytes 4-11) the token identifying the space to which the virtual address belongs to for entry A (page fault occurrence) of the appendage. In case of a data space, the token is the STOKEN returned by DSPSERV CREATE. Bytes 12 to 15 are

reserved. The same information (in the same area) has to be provided to the system on return from entry B (page fault completion). If the internal page fault queue is empty, an area of all zeros should be returned from entry B (page fault completion). This area has to be PFIXed.

If the AREA operand is not specified, the page fault information is contained in register 13; however, the appendage routine is not entered for a page fault of an address space with a space number larger than 255. The SETPFA SVC is canceled with illegal SVC if it is issued in an address space with a space number larger than 255.

DSPACE=NO | YES

Specifies whether the appendage routine is to process page faults for both address and data spaces (YES) or for address spaces only (NO). In this case, page faults caused by accessing data spaces are handled by the system and the appendage routine is not entered.

If DSPACE is specified, the AREA operand is required.

Following is a summary of linkage conventions which you must adhere to when you code an appendage routine.

General Coding Requirements

Your routine and all areas referenced by the routine must be fixed in real storage using the PFIX macro before the SETPFA is issued.

Register Usage

The following registers are used to pass information between the system and your appendage routine:

Register	Contents
7	The address for return of control to the supervisor of your VSE. Do not change the content of this register.
8	The address of your appendage routine. You can therefore use the register as the base register.
13	Information about the page fault to be handled (if the AREA operand is **not** specified):

Bytes	Contents
0-2	Shows the leftmost 24 bits of the 31-bit address of the page that is to be handled.
3	Space number

If the AREA operand is specified, the contents of register 13 should be ignored. The page fault information is then contained in the 16-byte area denoted by the AREA operand.

All other registers are undefined. However, the contents of all registers are saved by the supervisor before it passes control to your appendage routine.

Entry Linkage

This should be as follows:

```
       entryaddress  USING *,8
 (1)                 B     ENTRYA
 (2)                 B     ENTRYB
```

where:

(1) Used by the supervisor when a page fault occurs.
(2) Used by the supervisor after it has completed the handling of a page fault.

The appendage routine gets control in 24-bit addressing mode.

Page Fault Queue

Your routine must maintain a queue of 16-byte entries (if AREA is specified) or 4-byte entries (if AREA is not specified): one entry per private subtask controlled by your VSE task. Figure 50 on page 372 indicates how your appendage routine should manage the internal queue of page faults. Review Figure 50 after having read the remaining paragraphs about using the SETPFA macro. Note, however, that the page fault handling as outlined below does not apply if the AREA operand is specified. (If the AREA operand is specified, the 16-byte area - rather than register 13 - contains the page fault information.)

Processing in the Appendage Routine

This is described separately for the entry points ENTRYA and ENTRYB.

Control transfers to ENTRYA – **processing when a page fault occurs**. Do the following:

1. Examine the page-fault information.

   If you do not want to overlap handling of this page fault with your task processing, return control to the address in register 7 plus 4. This causes VSE to handle the page fault synchronously.

   Continue with the next step if you want to overlap page fault processing.

2. Store the contents of register 13 (if AREA is not specified) in the internally maintained page fault queue.
3. Set the task that caused the page fault into an internal wait state.
4. Dispatch a privately controlled subtask.

   You do this by storing, in bytes 8 through 15 of the task's save area, an EC-mode PSW. This PSW must contain the address of the subtask instruction to be executed first. To accomplish this:

   a. Set up this PSW (along with any values that should be loaded into certain registers for this subtask) in the save area of the subtask.
   b. Exchange this save area's contents with the contents of the save area for the task that issued the SETPFA macro.

   To be able to always dispatch a privately controlled subtask, provide one that merely issues a WAIT or a WAITM with an unposted ECB. Dispatch this subtask (as described above) if no other privately controlled subtask is dispatchable.

5. Return control to the address in register 7.

Control transfers to ENTRYB – **processing at the completion of a page fault**. Do the following:

1. Post the task that caused the now completed page fault so that it is removed from the internal wait.
2. Dequeue, from the internal page fault queue, the request that has just been handled.
3. Load the page fault information next in your queue, if any, into register 13 or set this register to zero if your queue is empty.
4. Return control to the address in register 7.

Processing for PF	Address in Register 13 From Superv.	Return to Superv. by	Internal Page Fault Queue	Address in Register 13 To Superv.
R1	PF address R1	BR 7	PF address R1	irrelevant
R2	PF address R2	BR 7	PF address R1 PF address R2	irrelevant
R3	PF address R3	BR 7	PF address R1 PF address R2 PF address R3	irrelevant
* any	any PF address	B 4(7)	PF address R1 PF address R2 PF address R3	unchanged
R4	PF address R4	BR 7	PF address R1 PF address R2 PF address R3 PF address R4	irrelevant
R1 complete	PF address R1	BR 7	PF address R2 PF address R3 PF address R4	PF address R2
R5	PF address R5	BR 7	PF address R2 PF address R3 PF address R4 PF address R5	irrelevant
R2 complete	PF address R2	BR 7	PF address R3 PF address R4 PF address R5	PF address R3

PF = Page fault
Rn = Page fault request n (1, 2, and so on) of your partition

* Page fault is not related to the partition.

Figure 50. Internal Page-Fault Queue and Communication with the System

SETPRT (Set the Printer) Macro

The SETPRT (printer setup) function serves as the interface for servicing printer setup requests between the following:

Job control and printer
Job control and operator
VSE/POWER and printer
User and printer
User and operator
User and VSE/POWER

Any request to alter the printer setup should be routed to SETPRT. SETPRT maintains a control block for each 3800 to reflect correct current printer status. Except for offset stacking and marking forms, application programs should not bypass the SETPRT routines and directly perform any of its services. This is because a subsequent SETPRT call, possibly in the next job, will not be able to correctly determine the machine's status.

The SETPRT routine is in the System Virtual Area (SVA) and is called by a macro issued by Job Control, VSE/POWER, or a problem program. When the device for which SETPRT was issued is a PRT1 printer and the FCB parameter was specified, the SETPRT routine issues an LFCB macro to use the appropriate FCB, and optionally, forms. When the device is a PRT1 but the FCB parameter was not specified, SETPRT performs no operation and gives a return code of zero. When the device type is not a 3800 or PRT1 printer, it is an invalid device type.

If the caller is running in the B-transient area and calls the SETPRT routine for a PRT1 printer, SETPRT performs no operation.

SETPRT issues one message to the operator defining necessary operator action (paper threading, forms change, or forms overlay frame change). SETPRT optimizes all requests for printer setup such that a setup request for a characteristic already established in the printer will not be performed again.

When a parameter is omitted from a call to SETPRT, the general principle is that SETPRT makes no changes to the corresponding characteristics for the printer. Exceptions are INIT and DFLT, and this concept does not apply to DCHK or SEP.

Before the SETPRT routine processes any parameter, except for WORKA, it determines whether the current SETPRT request will cause the device status to change. If a change will be needed, a Clear Printer command is issued, which fills the current page with blank lines if a partial page has been transmitted.

Any printer I/O errors encountered by SETPRT result in resetting the printer to IMPL status, writing an informational message to the printer, and setting a unique return code.

Although the caller of SETPRT identifies the printer with its logical unit identifier, the setup information is retained by the system according to the physical device address. This means that reassigning the printer does not cause a change to the setup status.

The SETPRT or QSETPRT macro calls the SETPRT routine. The macro generates a SETPRT parameter list and/or instructions to call the SETPRT routine. The macro exists in three forms controlled by the MF keyword:

- **Standard form**. Build a parameter list and then call the SETPRT routine (MF keyword omitted).
- **Execute form**. Point to an existing parameter list and then call the SETPRT routine (MF=(E,(address|(register)))).
- **List form**. Create a parameter list without any executable instructions and without linkage to SETPRT routine (MF=L).

Any parameter coded on an execute form macro overrides the same parameter coded on the corresponding list form of the macro.

The SETPRT and QSETPRT macros may use registers 0, 1, 14, and 15 in their expansions. The caller's addressability must be based on a different register.

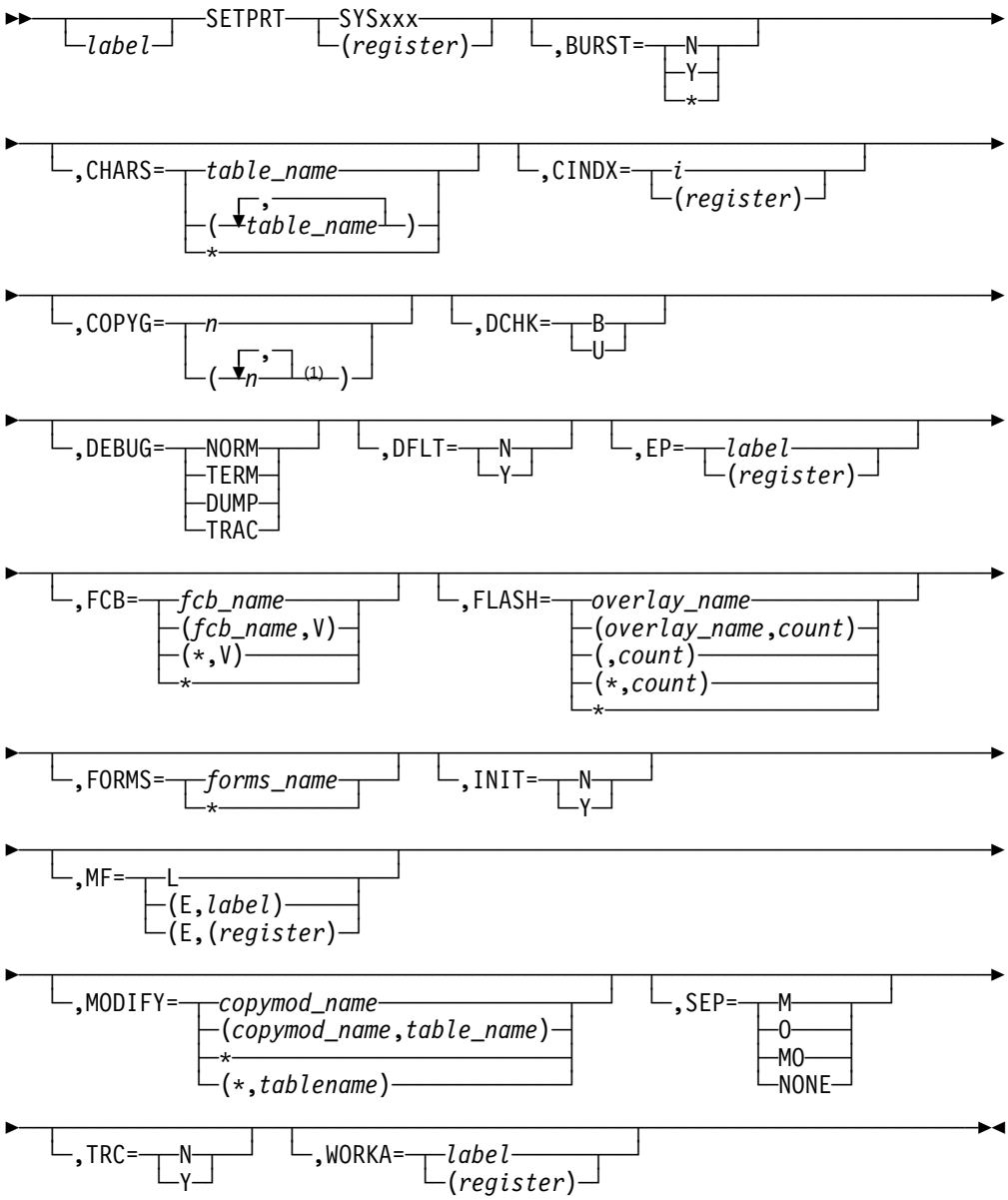
Any number of register value for the SETPRT and QSETPRT macros can be specified as an absolute (non-relocatable) expression. These are examples of valid absolute expressions:

12
X'C'
NUM (where NUM has been set by an EQU instruction to an absolute expression).
NUM+1

The advantage of using symbolic references is that the assembler cross reference shows other dependencies on the values. The disadvantage is that the macro cannot do as much checking of the validity.

The special values required for certain keywords, such as N, Y, V, B or L, must be specified as described.

The format of the SETPRT macro is:



Note:
[1] You can specify up to eight groups of copies.

label Optional label. If MF is omitted or coded as E, the name applies to the first executable instruction of the macro expansion. If MF=L is coded, the name applies to the beginning of the parameter list.

SYSxxx
Logical unit identifier for the printer to be set up. Only SYSLST and SYSnnn are supported. Instead of specifying the six-character name of the logical unit, the address of its LUB may be supplied in a register numbered in the range of 2 to 13. This parameter is always required by the SETPRT routine. It is required by the macro when MF is omitted. When MF is coded, the logical unit must be supplied in the execute form or list form of the macro.

BURST
Forms bursting request.

Y Specifies that the operator should thread the forms through the Burster-Trimmer-Stacker.

N Specifies that the operator should thread the forms to the continuous forms stacker.

***** The system default BURST setting is requested.

If the forms are not already threaded correctly, then message P300D is issued to ask the operator to change the threading.

If BURST is omitted, no change to the threading is requested.

CHARS

Character arrangement table names.

table name

Specifies the 1- to 4-character name of the character arrangement table (not including the system-assigned prefix, XTB1). Up to four names separated by commas and enclosed in parentheses, can be specified. No null value, such as CHARS=(AA,,BB) or CHARS=(,AA), can be specified. See Note on page 381.

***** The system default character arrangement table is requested. If this is specified, multiple *table names* are not allowed. If the operator has not specified a default for CHARS, the hardware default Gothic 10-pitch table is used.

CINDX

Transmission or copy group number.

i Is an absolute arithmetic expression. If it is not enclosed in parentheses, the value must be in the range of 0 to 255. If it is enclosed in parentheses, it identifies a register in the range of 2 to 13 containing a value from 0 to 255. Only the low-order byte of the register is used, but for compatibility with possible future support, it is recommended that the high-order three bytes be clear. How CINDX is interpreted depends on whether copy grouping (COPYG) is requested in the same call to SETPRT.

When copy grouping is not specified. CINDX specifies the transmission number. If CINDX is a number 1 through 255, it tells which copy (transmission) is about to begin. If zero is specified or no value is supplied to the SETPRT routine, then the copy number and grouping is not changed from the previous setting unless INIT=Y is specified. At the beginning of each job, there is no copy grouping.

When copy grouping is specified. The CINDX value identifies which copy group value to use. If CINDX is zero or not supplied to the SETPRT routine, then 1 is assumed. If the CINDX value is greater than the number of groups, COPYG is treated as if it were extended with group values of 1 each. The sum of the COPYG values and the values with which they are extended cannot exceed 255.

If both CINDX and COPYG are omitted, no change is made to the copy grouping or transmission count unless INIT is coded.

COPYG

Data set copy grouping information.

n Each *n* is an arithmetic expression specifying the number of copies of each page to be reproduced by the printer before printing the next page. If more than one *n* is specified, they are separated by commas and surrounded by parentheses. Each value can be from 1 to 255, except that if only one value is specified, it can be 0. Specifying 0 is mainly useful to clear a grouping when using the execute form. Up to 8 groups of copies can be specified. The sum of the group values cannot exceed 255. Which of the values is to be used for the current transmission is determined by the value specified or defaulted for the CINDX keyword. If both CINDX and COPYG are omitted, then the transmission number and copy grouping are not changed from their previous settings unless the INIT keyword parameter is also coded. When COPYG is coded and CINDX is not coded, a value of 1 is assumed for CINDX. See the CINDX description for the action taken when COPYG is omitted and CINDX is coded.

DCHK Block or unblock data check indicator.

B Specifies that data checks are to be blocked (prevented). This means that unprintable characters in the data printed after completion of SETPRT are to be treated as blanks.

U Specifies that data checks are to be unblocked (allowed). A data check is an I/O error.

If DCHK is omitted, data checks are blocked.

DEBUG

Select debugging options for SETPRT and QSETPRT errors. This can be useful for debugging errors by the caller or by the system. This option remains in effect until changed in a later SETPRT call or the end of the job.

NORM When an error is detected by SETPRT or QSETPRT, return to the caller with a return code in register 15, no matter what the return code is.

TERM Terminate the task with a CANCEL macro if a condition causing a return code greater than 4 in the low-order byte is detected. If it is the main (or only) task in the partition and OPTION DUMP was specified or defaulted in the JCL, a dump is taken to SYSLST. In the dump, register 15 contains the SETPRT or QSETPRT return code. Unless the return code is X'0028', X'2C', X'30', or X'38', a message is written to the printer.

DUMP Terminate program with dump if a condition causing a return code greater than 4 in the low-order byte is detected. In the dump, register 15 contains the SETPRT or QSETPRT return code. Unless the return code is X'0028', X'2C', X'30', or X'38', a message is written to the printer.

TRAC Initiate tracing of all significant SETPRT and QSETPRT events and if the return code is greater than 4, terminate partition with dump. If SYSLST is assigned, a dump of the SETPRT/QSETPRT work area is written to SYSLST at several points during execution. If SYSLST is not assigned, then DEBUG=TRAC has the same effect as DEBUG=DUMP. If SYSLST is assigned to a printer, each dump is preceded by a description of the dump.

If the DEBUG keyword is omitted, then the most recently specified value for DEBUG in a previous SETPRT (but not QSETPRT) in the job is taken. If DEBUG has not been specified during the job, DEBUG=NORM is assumed.

DFLT Establish printer defaults.

- **Y** Specifies that the printer is to be set with the defaults that were specified by the operator in the SETDF command. (See *VSE/ESA System Control Statements* under "SETDF".) It is equivalent to coding * for each of the parameters BURST, CHARS, FCB, FLASH, FORMS, and MODIFY that are not specified.
- **N** Is the default specification for this keyword and does not establish 3800 default setup.

EP Entry point address of the SETPRT routine. The EP keyword value can be a relocatable expression valid in an RX-type instruction or it can identify a register if it is in parentheses. If it is a relocatable expression, it must identify a word containing the address of the SETPRT routine. If a register is specified, the register must contain the entry point address of the SETPRT routine. The register can be 15 or in the range 2 to 13. EP cannot be coded when MF=L is coded. When EP can be coded but is not, the SETPRT macro generates code to obtain the address of the SETPRT routine.

FCB Forms control buffer information.

fcb name
Is the 1- to 4-character name of the FCB (not including the system-assigned prefix, FCB1). The length of the form defined by FCB must match the length of the actual forms loaded.

- **V** Indicates FCB verification. The FCB contents are formatted and printed on the 3800. Data checks are blocked and translate table zero is used for printing the FCB verification page.
- ***** The system default FCB is requested. If the operator has not specified a default FCB, the FCB loaded indicates 6 lines per inch with a channel-1 code defined on the first printable line, and the length set equal to that of the form currently loaded.

FLASH
Forms overlay or flashing request.

overlay name
Is the 1- to 4-character name of the forms overlay frame. If the specified frame is not already loaded, the operator is requested via message P300D to insert it in the 3800.

- **count** Is the number (from 0 to 255) of copies to be flashed, beginning with the first copy of the first transmission. If 0 is specified, the specified forms overlay frame is mounted but is not flashed. A specification of FLASH=(,count) means to flash the current forms overlay frame for the specified number of copies. If an *overlay name* but no *count* is specified, all copies are flashed.
- ***** The system default forms overlay is requested. If the operator has not specified a default, no flashing occurs.

FORMS

Paper forms request.

If the specified forms are not already loaded, then message P300D is issued to the operator requesting that the forms be changed. If the new form has a length different from the previous form and a new FCB is not specified, the 3800 loads the hardware default FCB. This can cause erroneous results later. To avoid this problem, either specify INIT or a new FCB when loading forms of a new length.

forms name

Is a 1- to 4-alphameric character forms identifier.

***** The system default form is requested. If the operator has not specified a FORMS default, form STANDARD is requested.

If the FORMS parameter is omitted, the operator is not asked to change the forms.

INIT Initializing the printer request.

Y Specifies that the printer be reset to hardware defaults of 6 lines per inch FCB, channel-1 code in the first printable line, a Gothic-10 folded character arrangement table, one copy, and no flashing. Copy modification and copy grouping are cleared. The burster threading is not affected. If TRC=Y is not also coded, then lines written to DTFs opened after this SETPRT will not contain TRCs (table reference characters). The TRC indicators in any open DTFs are not changed. The effect of INIT=Y is similar to the effect of the Initialize Printer channel command.

N Is the default and does not reset the printer to hardware defaults.

MF Identifies the form of the macro generation. Omission of this keyword generates the standard form with a parameter list and instructions to call the SETPRT routine.

(E,label) Identifies this as an execute form macro and specifies that the SETPRT parameter list at "label" is to be used for this SETPRT request. The parameter list is created from an L-form SETPRT macro or by execution of a QSETPRT macro. The parameter list must be on a word boundary and in the same protection key as the issue of SETPRT. The parameter list is updated with any parameters specified with this SETPRT macro except EP or WORKA. The name can be a relocatable expression of the form acceptable in an RX-type instruction, or a register in parentheses can be specified. The register must be in the range of 2 to 13, except that if EP is coded, then the register can be 1.

L Specifies that the list form of the macro instruction is used to create a parameter list that can be referenced by an execute form of the SETPRT macro instruction. The list is automatically aligned on a word boundary. No machine instructions are generated.

MODIFY

Copy modification information.

copymod name

Is the 1- to 4-character name of the copy modification phase (not including the system-assigned prefix, MOD1) to be loaded from the library into the 3800.

table name
Is the 1- to 4-character name of the character arrangement table to be used when printing the copy modification text. This character arrangement table need not be one of those specified or defaulted for CHARS if the 3800 has enough WCGMs installed. See Note on page 381. If *table name* is omitted, the first character arrangement table specified or defaulted with the CHARS keyword is used.

***** The system default copy modification is requested. If the operator has not specified a MODIFY default, any existing copy modification is eliminated.

If the MODIFY parameter is omitted, then the currently-loaded phase is used unless INIT=Y is also coded. Note that in some cases SETPRT must reload the currently-loaded copy modification into the printer even when you did not specify it. If it was obtained from a private library that is no longer assigned, SETPRT may give a return code that indicates that the copy modification phase could not be found even though it was not explicitly re-requested.

SEP Data set separation information. Omission indicates that no data set separation is required.

O Indicates that if the Burster-Trimmer-Stacker is being used, the 3800 should offset-stack the pages that follow from the pages that were previously transmitted. If the continuous forms stacker is being used, the 3800 changes the marking on the perforation edge from one line to two lines or vice versa.

M Indicates that, irrespective of whether offset stacking was requested, the current page is to be replicated three or five times and the perforations marked. The user can transmit a data set separator page prior to issuing SETPRT and the printer prints the page three or five times depending on the form length. The separator data must be contained within a single page and the 3800 must be positioned (relative to the FCB) within that page.

NONE Specifies that the separator options are to be reset (not used). This has the same effect as omitting the SEP parameter except that SEP=NONE can be used with MF=(E,label) to clear the SEP options in the parameter list.

TRC Table reference characters indicator.

N Indicates that for any DTFPR or DTFDI opened after this SETPRT, data lines do not contain table reference characters unless specified in the DTF macro. The table reference character will not be prefixed to each data line when presented to the access method.

Y Indicates that the first character of each data line given to the access method is a table reference character. This applies only to the issuance of PUT macros with DTFPR, DTFDI or DTFCP. The DTF must be opened after SETPRT is successfully issued.

WORKA
Work area address. This work area must be on a double-word boundary and at least 512 bytes or 12288 bytes long. When the caller provides a

work area, the first word must contain the length of the area, including the first word. The caller either identifies the work area or identifies a register name or number in parentheses. The register contains the address of the work area. The register must range from 2 to 13, unless EP is coded, in which case register 0 can be specified. The WORKA parameter cannot be specified at the same time as MF=L. When it is omitted, the SETPRT routine obtains storage space from the user GETVIS area and frees it before completing. If the partition does not have enough virtual storage space or is running in real mode, the caller must code the WORKA parameter.

If no valid work area is supplied (with the WORKA parameter), SETPRT obtains 512 bytes from the user GETVIS area. If the caller does not supply at least 12288 bytes and a character arrangement table, copy modification, or FCB is being processed, then an additional 11776 bytes are obtained from the user GETVIS area. The exception to this is that a SETPRT to a printer whose I/O is being trapped by VSE/POWER only requires 512 bytes for a work area. If an invalid work area is supplied, it is ignored and an area is gotten. All SETPRT-requested functions are performed using this area.

Note: The total number of character sets referenced by character arrangement tables in both the CHARS and MODIFY parameters cannot exceed the number of WCGMs available on the 3800 (either two or four). If the same character set is referenced by multiple character arrangement tables and graphic character modification is not used, then SETPRT loads only one copy of the character set. If a character set is referenced by two character arrangement tables and one is modified by graphic character modification and the other is not, then two character sets are loaded.

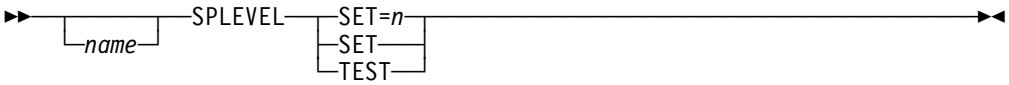
The SETPRT routine processes the parameters in the following order:

WORKA
SEP
INIT
TRC
BURST/FORMS/FLASH name
CHARS/MODIFY
FCB
COPYG/CINDX/FLASH count
DCHK

MF and EP are not relative to the SETPRT routine execution.

Certain SETPRT parameters have system-defined defaults available. These keywords are BURST, CHARS (first *table name* only), FCB, FLASH name, FORMS, and MODIFY (*copymod name* only). To request a system default, either code an "*" (asterisk) or omit the keyword and code DFLT=Y.

SPLEVEL (Set and Test Macro Level) Macro



The macro sets or tests the global symbol that indicates the level of a macro.

Certain macros supplied in the VSE/ESA 2.1 macro library are identified as being downward-incompatible with VSE/ESA 1.1, 1.2, or 1.3. In order to use these macros, you have to generate downward-compatible expansions by issuing the SPLEVEL macro. SPLEVEL sets (or tests) a global symbol that is interrogated by these macros during assembly to determine the type of expansion to be generated.

The following is a list of the macros that check the setting of the global symbol:

FCEPGOUT
PAGEIN
PFIX
PFREE
RELPAG
WTO
WTOR

SET=n
Sets a global symbol equal to n, where n must be 1, 2, 3, or 4. If you code any of the above macros, one of the following macro expansions is generated:

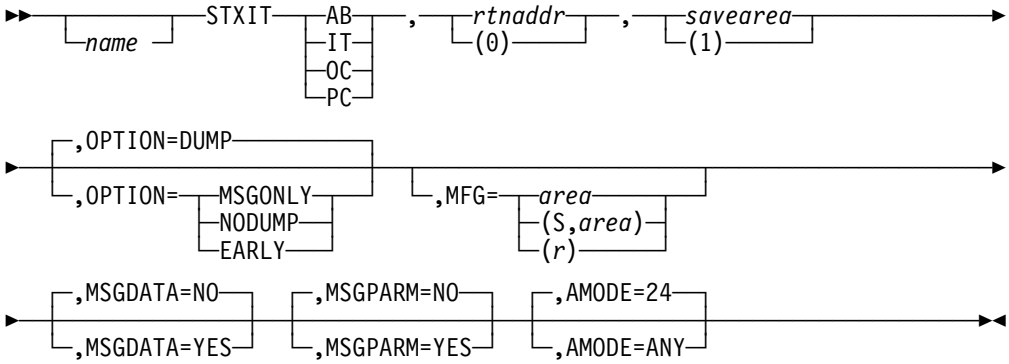
- VSE/ESA 1.1 and 1.2 macro expansion if n=1
- VSE/ESA 1.3 macro expansion if n=2 or 3
- VSE/ESA 2.1 macro expansion if n=4

SET Causes the SPLEVEL routine to use the default value n=4.

TEST Determines the macro level that is in effect. The result of the test request is returned to your program in the global set symbol &SYSSPLV. If TEST is specified and if SPLEVEL SET has not been issued during this assembly, the SPLEVEL routine puts the default value (4) into the global set symbol. If SPLEVEL has been issued, the previous value of n or the default value is already in the global set symbol. (For information about global set symbols refer to the High Level Assembler manuals.)

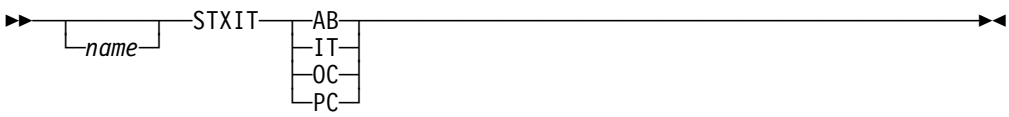
STXIT (Set Exit) Macro

To establish linkage:



MSGDATA and MSGPARM are valid for STXIT OC only.

To end linkage:



Requirements for the caller:

AMODE: 24 or 31

RMODE: 24 or ANY

ASC Mode: Primary

The macro establishes or ends linkage between the supervisor and an exit routine of your program for handling the specified condition. Linkage must be established before an interrupt occurs.

The STXIT macro is only allowed when the linkage stack is empty. If a STXIT macro is issued in a module called with a PC or BAKR instruction (that is, the linkage stack is **not** empty), the calling task is canceled.

The following types of exit routines can be specified:

AB Exit routine to handle abnormal termination conditions.

IT Exit routine to handle interval timer interrupts.

OC Exit routine to handle operator attention interrupts.

PC Exit routine to handle program checks.

Use the EXIT macro to return from your exit routine.

Warning

The STXIT macro should not be issued from an application running under CICS. Issuing this macro can adversely affect CICS execution.

The exit routine gets control:

- In the addressing mode specified in the AMODE operand of the STXIT macro; that is, in 24-bit addressing mode for AMODE=24 or in 31-bit addressing mode for AMODE=ANY.
- With the PSW key active at invocation time of the related STXIT macro or ATTACH macro.

When the STXIT macro is executed in 31-bit addressing mode, AMODE=ANY must be specified; otherwise a cancel condition (mode violation) occurs.

When you restart a program from a checkpoint, any STXIT linkages established prior to the checkpoint are destroyed.

If, in an exit routine, you are issuing an I/O request that requires the same logic module as your main routine, you must generate a read-only module by specifying RDONLY=YES in the DTF and in the logic module.

Both the main routine and the exit routine require a save area of their own.

Upon entry to an exit routine, the supervisor resets the PSW condition code, access register mode, and program mask to zero, and sets up the following registers:

- For STXIT AB, register 0 contains, in its low-order byte, the cancel code passed by VSE/ESA. For a list of these codes, see *VSE/ESA Messages and Codes*.
- Register 1 points to the user-supplied exit save area, which contains the interrupt status information and the contents of registers 0 through 15, plus the access registers (if the extended save area was used). For a detailed description of the information that is provided in the save area, see the “MAPSAVAR (Map Save Area) Macro” on page 309.
- Register 15 points to the exit routine entry (bit 0 being off indicates that the exit is in 24-bit mode; bit 0 being on indicates 31-bit mode).

All other registers have to be reloaded (for example, from the user-supplied save area) before they can be used. Do not rely on the contents of these registers at the time of interrupt; unpredictable results may occur if you do.

AB An abnormal task termination routine is entered if a task is terminated for a reason other than one of the following macros in the program: CANCEL, DETACH, DUMP, JDUMP, RETURN, and EOJ. (When OPTION=EARLY is specified, the AB exit routine is invoked for **any** type of termination, normal or abnormal.) Before invocation of the task's abnormal termination routine, the system produces termination messages and a partition dump depending on selected options (see the OPTION operand below).

The abnormal termination routine can then examine the interrupt status information (from register 1) and take whatever action is necessary.

Macros which might be used in this routine are, for instance, POST and CLOSE. The STXIT macro cannot be used; it would result in an abnormal termination (illegal SVC). Macros which should **not** be used are CHKPT, ENQ, LOCK, some I/O macros, and WAIT or WAITM in combination with

ECBs. Using these macros may result in an abnormal termination or a wait condition.

Note: An abnormal termination condition within the abnormal termination exit routine causes this routine to be terminated immediately. A deadlock situation may occur if a wait condition occurs within a subtask's abnormal termination exit routine that has to be posted by the main task.

After the appropriate action is taken, your abnormal termination routine may either resume processing using the EXIT AB macro (main task only) or terminate the task with CANCEL, DETACH, DUMP, JDUMP, EOJ, or with RETURN. For a main task, the whole job is terminated if OPTION=DUMP has been specified explicitly or by default. Only the current job step is terminated if OPTION=NODUMP and the termination macro used was DUMP or EOJ. Recovery via the EXIT AB macro is possible.

If your routine issues the DUMP or JDUMP macro, the system produces a storage map of the partition even if job control option NODUMP was specified.

Any task in a partition can attach a subtask with an ABSAVE operand in the ATTACH macro. This assumes the subtask will use the attaching task's abnormal termination routine. The length of the save area and the dump options are taken from the AB exit definition of the attaching task. If the ATTACH macro is issued in 31-bit mode, the length of the **extended** save area is assumed. However, the subtask may override the ABSAVE specification by issuing its own STXIT AB macro.

If an abnormal termination condition occurs in a main task and linkage has not been established to an abnormal termination routine, processing in the partition is abnormally terminated. However, if the abnormal termination condition occurs in a subtask without exit linkage, only the subtask is terminated.

When subtasks are detached or canceled, associated time intervals and exit linkages are cleared.

IT An interval timer interruption routine is entered when the specified interval elapses.

If an interval timer interrupt occurs while an interval timer exit routine is still processing, the handling of the interrupt is delayed. When processing ends with EXIT IT, the IT exit routine is entered again to process the new IT interrupt. (This can only occur if a short time interval was issued in your exit routine).

Note: If a task is using a logical transient routine when a timer interrupt occurs, your timer routine is not entered until the logical transient routine is released.

OC An operator communication interruption routine is entered when the operator enters the MSG command. Only the main task can issue the STXIT OC macro. Exception: In the VTAM partition, either the main task or a subtask can issue STXIT OC.

An operator communication interruption is ignored if no exit linkage has been established.

PC A program check interruption routine is entered when a program check occurs. If a program check occurs in a routine being executed from the logical transient area, the job containing the routine is abnormally terminated.

A program check interruption routine can be shared by more than one task within a partition. To accomplish this, issue the STXIT macro in each subtask with the same routine address but with separate save areas. To successfully share the same PC routine, the routine must be reenterable, that is, it must be capable of being used concurrently by two or more tasks.

If a program check condition occurs in a main task without exit linkage, processing in the partition is terminated. However, if this same condition occurs in a subtask, only the subtask is terminated.

The specified exit is not taken if either of the following is true:

- The program check occurs while VTAM is processing a request issued by the program.
- The STXIT macro was issued with a PSW-key not equal to 0 and the PSW-key at the time of the program check is not equal to the PSW-key when the STXIT macro was issued.

rtnaddr

Specifies the entry point address of the routine that processes the condition described in the first operand. Your exit routine may be located anywhere in the program if AMODE=ANY. If AMODE=24, it must be located below the 16MB line.

Note: If the routine address is zero, the termination function is executed.

savearea

Specifies the address of a save area in which the supervisor stores the interrupt status information. Your program must have a separate save area for each routine that is included. The save area may be located anywhere in storage if AMODE=ANY; if AMODE=24, it must be located below the 16MB line. For the format of the save area, see the “MAPSAVAR (Map Save Area) Macro” on page 309.

OPTION=DUMP | MSGONLY | NODUMP | EARLY

This operand can be used only when setting up linkage to an abnormal termination exit routine (STXIT AB). The effect of the various specifications is as follows:

OPTION=DUMP (or omission of the operand)

Before the abnormal termination routine receives control, the system issues termination messages. In addition, the system produces a partition dump, unless the job control option NODUMP is active.

OPTION=MSGONLY

Before the abnormal termination routine receives control, the system issues termination messages. No dump is produced in this case.

OPTION=NODUMP

Neither a termination message is issued nor a dump is produced. However, if the abnormal termination routine terminates abnormally, termination messages and the dump are given regardless of this OPTION specification.

Note: If your routine ends with a DUMP macro and the STXIT macro was specified without OPTION=NODUMP, you get two dumps.

OPTION=EARLY (applies to subsystems only)
This causes the AB exit routine to be invoked for any type of termination (normal or abnormal) and, for a main task, before propagating the termination to its subtasks.

An exit with OPTION=EARLY can be set up only once during the whole lifetime of a task. Any subsequent request to modify or reset this exit is ignored. This protects the early exit from being overwritten by any user code that is executed under the same task as the subsystem.

For an STXIT request with OPTION=EARLY, the system sets one of the following return codes into register 15:

X'00' Exit successfully set.
X'04' Exit already set.
X'08' Reset not allowed.
X'0C' No subsystem request.

An AB exit routine defined with OPTION=EARLY cannot be transferred via the ATTACH (ABSAVE=) macro. If an AB exit is to be transferred to the attached task, a secondary AB exit (other than OPTION=EARLY) must be defined either prior to or after the STXIT AB OPTION=EARLY macro request (the AB exit becomes secondary when the STXIT OPTION=EARLY is issued). The secondary AB exit is only used in the ATTACH processing.

MFG=area | (S,area) | (r)
This operand is required if the program that issues the STXIT macro is to be reenterable. (It is not required if STXIT is issued in 24-bit mode, and **rtnaddr** plus **savearea** are specified in register notation, and no other operand is specified.)

The operand specifies the address of a 64-byte dynamic storage area, that is, storage which your program obtains through a GETVIS macro. The area is needed by the system during execution of the macro. Registers 0 and 1 may not be used for register notation.

MSGDATA=NO | YES
MSGDATA=YES indicates that the operator communication (OC) exit is prepared to retrieve data from the AR MSG DATA command. In that case VSE assumes that the *extended* save area is used. The layout of the area is defined with the mapping macro MAPSAVAR. If the STXIT macro was issued with AMODE=24, the MAPSAVAR fields SVUAPSW (actual PSW) and SVUAREG (access register save area) are not used.

If no data is specified with the MSG command, the area is initialized with binary zeros.

MSGDATA=NO indicates that no data is to be passed from the MSG command and that the DATA operand, if specified, is to be ignored.

MSGDATA and MSGPARM are mutually exclusive.

MSGPARM=NO | YES
Indicates that routing and correlation parameters associated with the MSG command are to be passed to the OC exit in the user save area, along with

a pointer to MSG data, if any (see the “MAPSAVAR (Map Save Area) Macro” on page 309). MSGPARM and MSGDATA are mutually exclusive.

If MSGPARM=YES is specified on STXIT OC, the scheduled save area is used. The following fields are added to MAPSAVAR in the SVUMGADR area, and used to pass MSG parameters to the OC exit for MSGPARM=YES:

SVUMCSID
4-byte console ID (CONSID) of the console where the MSG command was entered. The setting of the high order bit indicates if the console has 'master' (0) or 'user' (1) authority.

SVUMNAME
8-byte name of the console where the MSG command was entered.

SVUMCART
8-byte command and response token (CART) associated with the MSG command.

SVUMDLNG
2-byte length of MSG data.

SVUMDATA
31-bit pointer to MSG data (zero if no data specified).

To ensure correct routing and correlation of all messages generated by the OC exit, such messages must be written via WTO/WTOR, with CONSID and CART parameters matching the values passed in SVUMCSID and SVUMCART respectively.

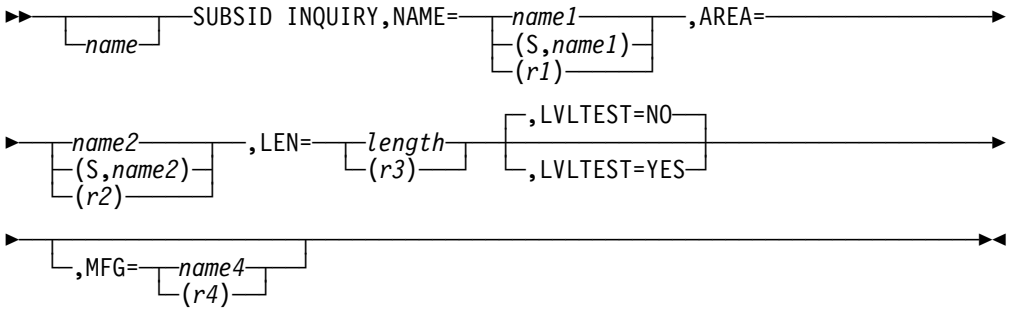
Use of the MSG command from a user console is restricted by the system to partitions running under control of that console (ECHO option).

AMODE=24 | ANY
Specifes the type of exit save area that is to be used for status saving and the addressing mode in which the exit routine gets control. AMODE=24 indicates that the old (72-byte) save area is to be used and that the exit routine is to get control in 24-bit addressing mode. AMODE=ANY indicates that the *extended* save area (with access registers) is to be used and that the exit routine is to get control in 31-bit addressing mode. For the layout of the save areas, see the “MAPSAVAR (Map Save Area) Macro” on page 309.

Figure 51 shows what happens when one of the four exit conditions occurs while an STXIT routine is being processed within a particular partition.

Figure 51. Effect of an AB, IT, OC, or PC Interrupt During STXIT Routine Execution

SUBSID (Subsystem Information Display) Macro



Requirements for the caller:

AMODE: 24 or 31

RMODE: 24 or ANY

ASC Mode: Primary

The macro allows you to make inquiries about the supervisor. The information about the supervisor (such as version number, modification number, or some indicators) is described by a byte string, which may be interpreted with the help of the macro MAPSSID.

NAME=name1 | (S,name1) | (r1)
Specifies the address of a 4-byte field containing the name SUPb (where b = blank).

AREA=name2 | (S,name2) | (r2)
Specifies the address of the area into which the requested information is to be stored.

LEN=length | (r3)
Specifies the length of the area as an integer, a self-defining term, or as a value in a register. The length to be specified can be obtained from the DSECT generated by the MAPSSID macro.

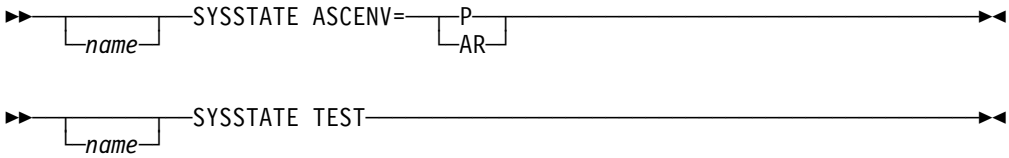
LVLTEST=NO | YES
Specify LVLTEST=YES if the program might make the inquiry on a pre-VSE release (DOS/VS) supervisor that does not support the SUBSID function. This prevents the program from being canceled. If you specify LVLTEST=NO an inquiry under these circumstances causes the program to be canceled.

MFG=name4 | (r4)
The operand is required if the program is to be reenterable. It specifies the address of a 64-byte dynamic storage area, that is: storage which your program obtained through a GETVIS macro. This area is required for system use during execution of the macro.

Return Codes in Register 15

0 Information returned.
8 Returned information truncated, because the area specified is too short. Register 0 contains the total length in the 2 rightmost bytes.
16 Name not found.
20 SUBSID function not available because this is a back-level supervisor (only if LVLTEST=YES).

SYSSTATE (Set and Test Address Space Control Mode) Macro



Requirements for the caller:

AMODE: 24 or 31

RMODE: 24 or ANY

ASC Mode: Primary

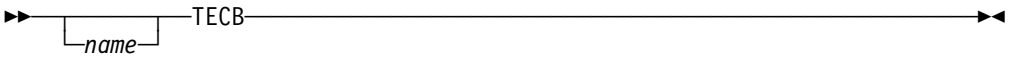
The SYSSTATE ASCENV macro sets a global symbol that indicates whether your program is running in AR (access register) mode or in primary ASC (address space control) mode. This information is needed by certain macros (DSPSERV, SDUMP, SDUMPX, for example) that support callers in both AR and primary mode and that need to know in which mode your program is running. The macros that support callers in AR mode generate code and addresses that are appropriate for AR mode; macros that support callers in primary mode generate code and addresses for primary mode. These macros use the SYSSTATE TEST option to test the global symbol that was set with the SYSSTATE ASCENV macro.

Issue the SYSSTATE ASCENV=AR macro at the time your program changes ASC mode to AR mode. Then, when your program returns to primary mode, issue SYSSTATE ASCENV=P.

ASCENV=P | AR
Specifies whether your program is running in primary (P) or access register (AR) mode.

TEST Determines the current mode by checking the global symbol that was set by the most recent invocation of SYSSTATE ASCENV. Depending on the setting of the global symbol, the caller of SYSSTATE TEST generates code and addresses appropriate for primary mode or AR mode.

TECB (Timer Event Control Block) Macro

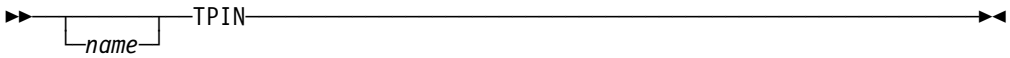


Required RMODE: **24** or **ANY**

The macro generates a timer event control block which can be referred to by the symbol you specify in the name field. This block contains an event bit that indicates when the time interval specified in SETIME has elapsed. The format of this block is as follows:

Bytes	Purpose of Bits	
0-1	Reserved.	
2	Control byte:	
	0	If 0: the time specified in SETIME has not elapsed. If 1: the time specified in SETIME has elapsed.
	1-7	Reserved.
3	Reserved.	

TPIN (Telecommunication Priority In) Macro



Requirements for the caller:

AMODE: 24

RMODE: 24

ASC Mode: Primary

The macro is available primarily for telecommunication applications that require immediate system response. The macro causes one or more partitions (other than the one issuing the macro) to be deactivated. The number of partitions that can be deactivated is specified at the console by way of a TPBAL command. The partitions to be deactivated are the ones with the lowest priorities.

This request is ignored in each of the following cases:

- The operator has not made TP balancing active by means of the TPBAL command.
- None of the partitions specified in the TPBAL command contains a program running in virtual mode.
- The only partition that could be affected by TP balancing is the partition that issued the TPIN request.
- There is no paging in the system.

The TPIN macro must always be used in conjunction with the TPOUT macro. The operand field is ignored.

In a system without page data set, the macro results in a null operation.

TPOUT (Telecommunication Priority Out) Macro

Requirements for the caller:

AMODE: 24

RMODE: 24

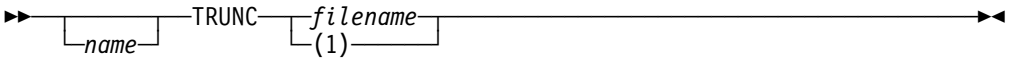
ASC Mode: Primary

The TPOUT macro causes the system to reactivate partitions that had been deactivated by the TPIN macro.

Failure to issue the TPOUT macro can cause considerable and unnecessary performance degradation in the batch partition(s). The operand field is ignored.

In a system without page data set, the macro results in a null operation.

TRUNC (Truncate Block) Macro



Requirements for the caller:

AMODE: 24

RMODE: 24

ASC Mode: Primary

The macro is used with blocked output records written to disk or to magnetic tape. It allows you to write a short block of records. These short blocks do not include padding. Thus, the macro can be used for a function similar to that of the RELSE macro for input records. When the end of a category of records is reached, the last block can be written and the new category can be started at the beginning of a new block.

The TRUNC macro does not necessarily cause a physical write to an FBA disk. If this is desirable, code the PWRITE operand in the DTFSD macro for the affected file.

filename | (1)
The symbolic name of the file specified as name in the DTFxx macro for the affected file.

TTIMER (Test Interval Timer) Macro

Requirements for the caller:

AMODE: 24 or 31

RMODE: 24 or ANY

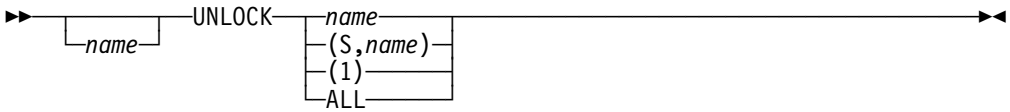
ASC Mode: Primary

The macro is used to test how much time has elapsed of an interval which was set in the same task by the associated SETIME macro. The TTIMER macro returns the time remaining of the interval, expressed in hundredths of seconds in binary, in register 0.

CANCEL
If this is specified, the time interval set in that task is canceled. As a result, the interval timer interruption routine of the task (to which linkage may have been established by an STXIT IT macro) does not receive control. If the corresponding SETIME macro specified the same name of a TECB, that TECB's event bit is set to 1.

If you omit the operand, you can include a comment in the macro only if this comment begins with a comma.

UNLOCK (Unlock Resource) Macro



Requirements for the caller:

AMODE: 24 or 31

RMODE: 24

ASC Mode: Primary

The macro can be used to:

- Dequeue the issuing task (or partition) from the named resource (to which the task had previously been queued by a LOCK macro). The resource must have been defined to the system by a DTL or GENDTL macro.
- Lower the lock control level. This may be done only if the issuing task is currently locked, onto the resource, with the most stringent control level: CONTROL=E and LOCKOPT=1 (the CONTROL and LOCKOPT operands are described with the DTL macro). The resource then continues to be held by the task. However, another task waiting for this resource can be dispatched again and may gain shared access (see also the description of the “LOCK (Lock a Resource) Macro” on page 299). To use the UNLOCK macro for this purpose, you must issue the MODDTL macro with CHANGE=ON.

name | (S,name) | (1)
Specifies the DTL address.

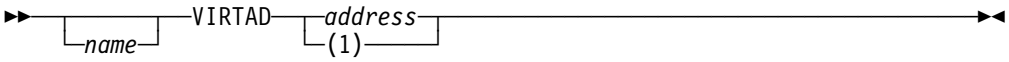
ALL Frees all resources which are locked by the task and whose DTLs were defined with KEEP=NO. If UNLOCK ALL is issued by the main task, not only the resources locked by that task are unlocked, but also those which have been locked by subtasks, with OWNER=PARTITION specified for DTL generation.

Return Codes in Register 15

0 Successful request; the resource has been unlocked.
4 The resource is not locked for the unlocking task.
8 DTL format error.

Note: UNLOCK ALL does not provide a return code, and register 15 remains unchanged.

VIRTAD (Virtual Address Return) Macro



Requirements for the caller:

AMODE: 24 or 31

RMODE: 24 or ANY

ASC Mode: Primary

The macro returns, in register 0, the virtual address corresponding to a specified real address.

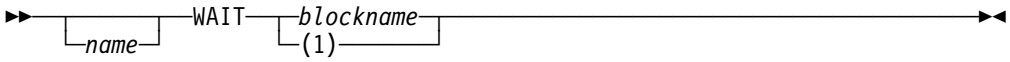
address | (1)
Is the real storage address to be converted. It can be given as a symbol or in register notation.

The AMODE/RMODE of the caller can be 24 or 31 bit. When called in 24-bit mode, the address will be treated as a 3-byte address; when called in 31-bit mode, the address will be treated as a 4-byte address.

Register 0 returns the virtual address only if the specified real address points to a page frame that contains a PFIXed page. Otherwise register 0 contains 0. Thus, the macro can be used to test if a page is PFIXed.

Note: The pages of a program running in real mode are considered to be fixed.

WAIT (Wait for Event) Macro



Requirements for the caller:

AMODE: 24 or 31

RMODE: 24 or ANY

ASC Mode: Primary

Issue this macro whenever your program requires that an event is completed before processing by the program continues. You need the macro, for example, if an I/O operation (started by an EXCP macro) must be completed before processing continues.

With the WAIT macro, a task sets itself into the wait state until the event control block (ECB) specified in the macro is posted. An ECB is posted if its event bit (bit 0 of byte 2) is set to 1.

WAIT processing can wait for an ECB in 24-bit or 31-bit addressing mode physically resident above or below 16MB. When WAIT is issued in AMODE 24, the ECB address is treated as 24-bit address. When WAIT is issued in AMODE 31, the ECB address is treated as 31-bit address.

The types of ECBs that may be referred to in the macro are any of the following:

- User defined (by a DC F'0' or as the operand of a macro such as ATTACH).
- TECB
- CCB or IORB (24-bit address only).

For a discussion of the use of ECBs, see “Specifying an Event Control Block” in the *VSE/ESA System Macro User's Guide*; for a description of the TECB, CCB, or IORB see the corresponding macros.

ECBs are normally used to synchronize tasks within the same partition. Use the XPCC support when tasks belong to different partitions.

Do not use the macro to wait on any of the ECBs listed below if they are not associated with the task (for instance, elapsed timer interval or an I/O operation not started by the same task).

The ECBs are:

Telecommunication ECB
TECB
CCB or IORB (24-bit address only).

An ECB is posted in either of the following ways:

- Automatically by the system if the macro requesting the involved service included a valid ECB specification. Examples are:

  The CCB or IORB referred to in an EXCP macro.
  The ECB referred to in an ATTACH macro.

- By your program when it issues a POST macro referring to the ECB.

Do not use the WAIT macro together with logical IOCS request macros such as GET, PUT, or CNTRL.

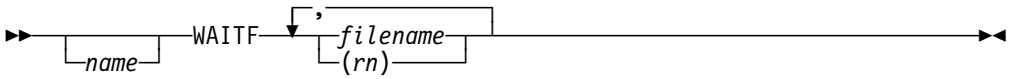
When a WAIT macro is processed, and the corresponding event control block is not posted, the issuing task is set into the wait state. Control is then passed to the supervisor, which makes the processor available to another task in the same or in another partition.

Notes:

1. When a WAIT macro is processed and the corresponding event bit is on, the task retains control. If an ECB or a TECB is to be used more than once, it is the task's responsibility to reset the event bit as soon as possible after it has been posted.
2. Telecommunication ECBs must not be waited for, because their format does not satisfy a WAIT.

blockname | (1)
The name of the event control block or the CCB or IORB, specified as a symbol or in register notation. For a CCB or an IORB, this is the name of the control block used in the EXCP macro.

WAITF (Wait for Completion of I/O) Macro



Requirements for the caller:

AMODE: 24

RMODE: 24

ASC Mode: Primary

The macro ensures that the transfer of a record is complete. It is valid for DAM and ISAM; but for SAM only with MICR and OCR devices.

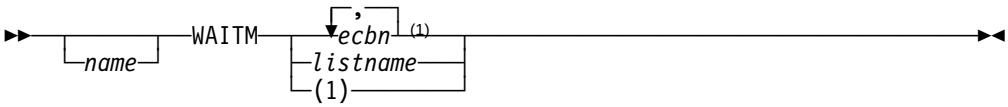
The WAITF macro is issued after any READ or WRITE for a file and before the succeeding READ or WRITE for the same file. If the I/O operation is not completed when WAITF is issued, the partition is placed in a wait state until the data transfer is completed. This allows processing of programs in other partitions while waiting for completion. When data transfer is complete, and if no errors were encountered, processing continues with the next sequential instruction. If an error is encountered, control passes to the error-handling routine named in the DTFxx macro.

filename|(r1),filename|(r2),...

For filename, specify the name you used in the DTFxx macro for the file from which a record is being read or to which a record is being written. You may specify this name as a symbol or in register notation. Multiple file names are valid only when using SAM to read MICR records.

If you are using the multiple-file-name format of the macro while processing MICR records, and if any of the files have records or errors ready to be processed, control remains in the partition and processing continues with the instruction following the WAITF.

WAITM (Wait for Multiple Events) Macro



Note:
[1] You can specify up to 16 ECB's.

Requirements for the caller:

AMODE: 24 or 31

RMODE: 24 or ANY

ASC Mode: Primary

The macro enables your program or task to wait for one of a number of events to occur. Control returns to the task when at least one of the event control blocks specified in the WAITM macro is posted. Refer to the “WAIT (Wait for Event) Macro” on page 397 for a description of the types of event control blocks and the restrictions on their use.

WAITM processing can wait for ECBs in 24-bit or 31-bit addressing mode physically resident above or below 16MB.

- When WAITM is issued in AMODE 24, the passed ECB addresses or list address are treated as 24-bit addresses. The *first byte* following the last address in the list must be nonzero to indicate the end of the list.
- When WAITM is issued in AMODE 31, the passed ECB addresses or list address are treated as 31-bit address. The *first bit* of the last address in the list must be nonzero to indicate the end of the list.

On return, register 1 holds a posted entry of the ECB list. If register 1 holds the last list entry, the high-order bit is set (only for user-defined ECB lists).

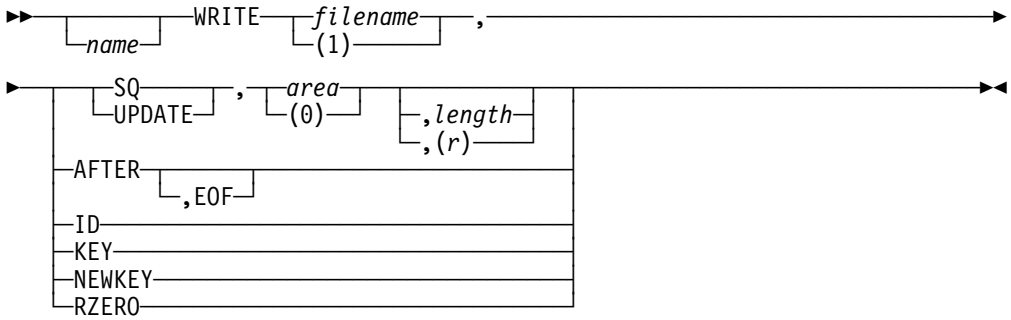
When control returns to a waiting task, register 1 points to the posted event control block (byte 2, bit 0 set to 1).

Note: Telecommunication ECBs must not be waited for, because their format does not satisfy a WAITM.

ecb1,ecb2,... | listname | (1)
The operand provides the addresses of the ECBs to be waited upon. The names of ecb1, ecb2... are assumed when at least two operands are supplied. Up to 16 names can be coded.

If only one operand is supplied, it is assumed to be the name (listname) of a list of consecutive fullword addresses that point to the ECBs to be waited upon.

WRITE (Write a Record) Macro



Requirements for the caller:

AMODE: 24

RMODE: 24

ASC Mode: Primary

The WRITE macro transfers a record from virtual storage to an output file.

filename | (1)
For filename specify the same name you used as name in the DTFxx macro for the file. Register notation must be used if your program is to be self-relocating.

SQ | UPDATE
For sequential work files, specify SQ for magnetic tape work files. For disk work files, specify:

SQ for a formatting write.
UPDATE for a non-formatting write.

When writing to an FBA disk, a non-formatting WRITE (with UPDATE) writes the current CI (control interval), while a formatting WRITE (with SQ) writes the CI and follows it immediately with a Software-End-Of-File (SEOF).

When writing to a CKD disk, a formatting WRITE writes count, key, and data, while a non-formatting WRITE writes only data.

area | (0)
For a sequential file, this specifies the name, as a symbol or in register notation, of the output area used by the file.

length | (r)
Specifies the actual number of bytes to be written on a sequential file. Determines only records of undefined format (RECFORM=UNDEF).

AFTER
For a DA file, specify AFTER to write a record after the last record written, regardless of key or identifier.

EOF Applies only to the WRITE...AFTER form of the macro. Specify EOF to write an end-of-file on a track after the last record on the track, if this is desired.

ID For DA files, specify ID to write at a location determined by the record identifier in the count area of the records.

KEY For an indexed sequential file, specify KEY for random updating. For a direct access file, specify KEY to write at a location determined by the record key (control information is in the key area of the records).

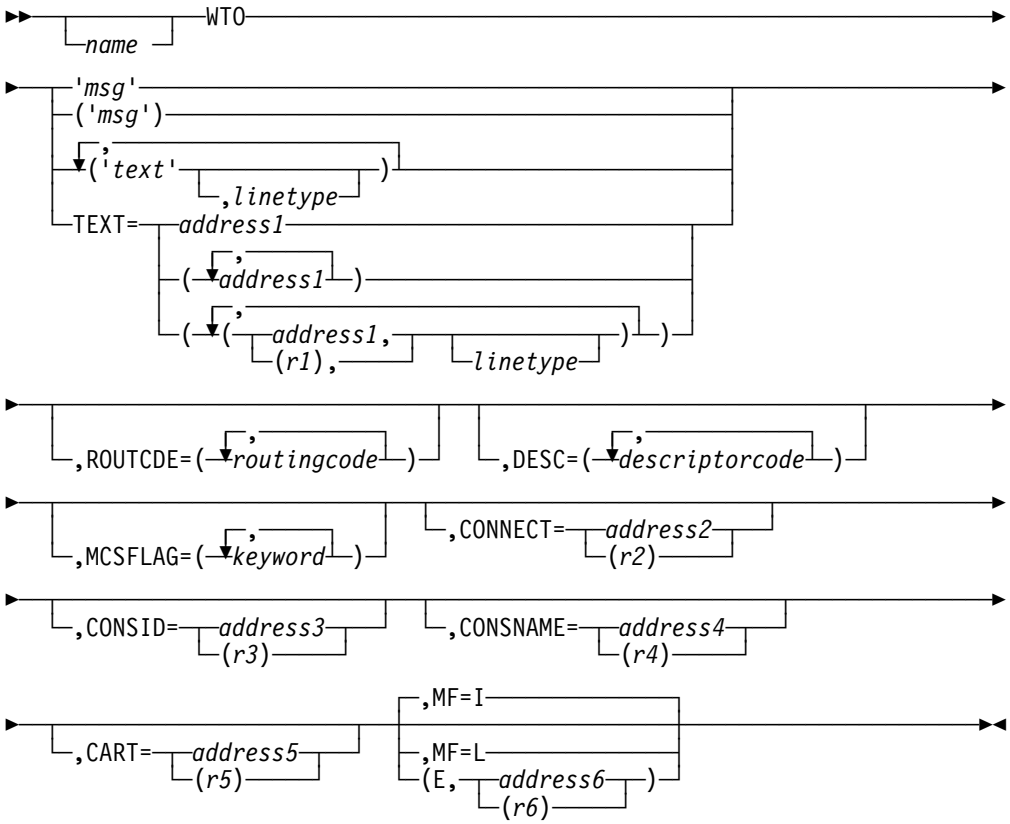
NEWKEY
Applies only to indexed sequential files. Specify NEWKEY to write a new (not updated) record in the file.

When loading or extending the file, precede the WRITE filename,NEWKEY with a SETFL macro and follow it with an ENDFL macro.

When adding a record after sequential retrieval, issue an ESETL macro before writing the record.

RZERO
For a direct access file, specify RZERO to reset the capacity record of a track to its maximum value and erase the track after record zero.

WTO (Write to Operator) Macro



Except where noted otherwise, only registers 2 to 12 may be used for register notation.

Requirements for the caller:

AMODE: 24 or 31

RMODE: 24 or ANY

ASC Mode: Primary

The WTO macro is used to issue a console message. It allows to use message text addresses (TEXT operand) in addition to literal text, to issue log-only messages (MCSFLAG operand), to use multiple WTOs for a single message (CONNECT operand), to direct a message (typically a command response) to a specific console (CONSID operand), and to correlate command responses with the original command (CART operand).

Both source and object level compatibility is preserved for programs using the WTO version of earlier VSE/ESA releases. However, the VSE/ESA 2.1.0 default expansion is backward incompatible. If none of the 2.1.0-unique operands is used, a backward-compatible expansion, suitable for execution on all VSE/ESA releases, may also be obtained with the VSE/ESA 2.1.0 WTO macro, by issuing SPLEVEL SET=n with n < 4 prior to WTO invocation (see also “SPLEVEL (Set and Test Macro Level) Macro” on page 382).

'msg' | ('msg')
Denotes a single-line literal message. The message text must be enclosed in quotes and may be up to 125 characters long. Quotes to be included in the message text must be coded as double quotes. All characters with a value smaller than X'40' (except for the DBCS control characters SI and SO) or equal X'FF' are replaced by blanks.

('text',linetype)...
Denotes a message consisting of up to 10 lines. The text of each line must be enclosed in quotes, its maximum length depends on the specified line type (see below). Quotes to be included in the message text must be coded as double quotes. All characters with a value smaller than X'40' (except for the DBCS control characters SI and SO) or equal X'FF' are replaced by blanks.

linetype may be specified as one of the following:

C Denotes a control line containing a message title. It can only be specified for the first message line. The maximum text length is 34 characters.

L Denotes a label line containing message heading information. It can be specified for at most two consecutive lines immediately following the C line, if any. The maximum text length is 70 characters.

D Denotes a detail message line. Up to ten detail lines can be specified, each with a maximum of 70 characters. This is the default when the line type is omitted and two or more lines are specified.

DE Denotes the last detail line and, in addition, the end of the whole message. Only one DE line with a maximum length of 70 characters may be specified. DE is implied for a single line message, when linetype is omitted.

E Indicates that the previous line was the last message line. E is mutually exclusive with DE. Only one E line may be specified. The text portion should be omitted for this line and is in any case ignored.

Incomplete multi-line messages not terminating with a linetype of DE or E and to be continued on subsequent WTO macros with the CONNECT operand, are only supported for known subsystems or authorized vendor products. Otherwise, DE is enforced for the last message line, or an E line is appended by the system.

TEXT=...
Denotes a message consisting of up to 10 lines. The text of each line is specified by the address of an area containing a 2-byte length, followed by a character string of that length. The maximum length for a single line with line type omitted is 125 characters, and depends otherwise on the line type. All characters with a value smaller than X'40' (except for the DBCS control characters SI and SO) or equal X'FF' are replaced by blanks.

The line type is used as described above.

ROUTCDE=(routing code...)

Specifies one or more routing codes as decimal numbers in the range 1-128, separated by commas or by a hyphen to indicate a range. When ROUTCDE is omitted, a default routing code of 2 is assumed. ROUTCDE is ignored when specified together with CONNECT (see below).

Routing codes determine, along with the CONSID or CONSNAME operands (see below) and the VSE/POWER JECL JOB ECHO option, on which console(s) the message is to be displayed and whether or not the message is to be logged on the hardcopy file. The meaning associated with each routing code is as follows:

1 Master Console Action: The message indicates a change in the system status and demands action by an operator with master authority.

2 Master Console Information: The message indicates a change in the system status and informs about a condition that might require action by an operator with master authority.

3 Tape Pool: The message gives information about tape devices such as the status of a tape unit or a request to mount a volume.

4 Direct Access Pool: The message gives information about DASD devices such as the status of a DASD unit or a request to mount a volume.

5 Tape Library: The message gives tape library information such as mount requests qualified by volume serial numbers.

6 Disk Library: The message gives disk library information such as mount requests qualified by volume serial numbers.

7 Unit Record Pool: The message gives information about unit record devices such as a request to mount a printer train.

8 Teleprocessing Control: The message gives information about teleprocessing equipment such as a notification of line errors.

9 System Security: The message gives information about security checking such as a request for a password.

10 System Error/Maintenance: The message gives problem information for the system programmer such as a system error, an uncorrectable I/O error, or information about system maintenance.

11 Programmer Information: The message is intended for the problem programmer and is to be routed to the console identified by a VSE/POWER JECL JOB ECHO option or to an ICCF terminal.

12 Reserved for emulators.

13-20 Reserved for customer use.

21-28 Reserved for subsystem use.

29-64 Reserved for IBM use.

65-96 Reserved for messages associated with particular processors.

97-128
Reserved for messages associated with particular devices.

When WTO is issued from an ICCF interactive partition and only routing code 11 is specified, the message is only displayed on the ICCF terminal and is not logged on the HC file. If other routing codes are (also) present, the message is (also) routed and logged according to the general rules.

DESC=(descriptor code...)
Specifies one or more descriptor codes as decimal numbers in the range 1-16. Codes 1 to 6, 11 and 12 are mutually exclusive. If more than one of these codes are specified, the most significant one is used (see below), the others are ignored. Code 7 can be assigned in combination with any other code. When DESC is omitted, a descriptor code of 7 is assumed. DESC is ignored if specified together with CONNECT (see below).

Descriptor codes determine, along with other factors, the presentation and retention attributes of a message. The meaning associated with each descriptor code is as follows:

1 System Failure: The message indicates an error that disrupts system operations. To continue, the operator may have to re-IPL the system or restart a major subsystem.

2 Immediate Action Required: The message indicates that the operator must perform an action immediately. Some tasks may be in a wait state until the action is performed, and system performance is affected.

3 Eventual Action Required: The message indicates that the operator must perform an action eventually. No tasks are waiting for action completion.

4 System Status: The message indicates the status of a system task or of a hardware unit.

5 Immediate Command Response: The message is issued as a response to a system command.

6 Job Status: The message indicates the status of a job or job step.

7 Task-Related: The message is related to the processing of an application or system program and is automatically DOMed by the system when the related job step ends after logging, if applicable.

This descriptor code is assigned automatically when the message is issued by a user task on its own behalf.

8-10 Not used by VSE, ignored when specified.

11 Critical Eventual Action Required: The message indicates that the operator must perform an action eventually, and no tasks are waiting for action completion. However, the action is important enough for retaining the message on the console screen until the action is completed.

12 Important Information: The message contains important information that must be displayed at a console, but does not require any action in response.

13-16 Reserved.

MCSFLAG=(keyword...)
Specifies one or more keywords requesting some special handling for this message. Only the following specifications are supported by VSE/ESA:

RESP The message is a command response.

HRDCPY
The message is only to be logged on the hardcopy file. All routing parameters (ROUTCDE, CONSID, CONSNAME) are ignored.

BUSYEXIT
Shortage of message buffers is to be handled by a return code, rather than by waiting for buffers to be freed.

CONNECT=address2 | (r2)
Specifies (for privileged applications only) the address of a 4-byte field or a register containing a message ID returned by a previous WTO to which this WTO is to be connected. Connected WTOs are treated as one logical message and only the last line of the last WTO must be an E or DE line. CONNECT to or for a single line message, with linetype omitted, is invalid.

CONNECT is mutually exclusive with CONSID, CONSNAME and CART. Also, a connected message inherits routing and descriptor codes from the message it is connected to. Therefore, ROUTCDE and DESC operands specified together with CONNECT are ignored.

CONSID=address3 | (r3)
Specifies the address of a 4-byte field or a register containing the ID of the console to which this message is to be directed.

CONSID is invalid when CONNECT is specified. In this case, the console ID of the first WTO applies to all connected WTOs.

CONSNAME=address4 | (r4)
Specifies the address of an 8-byte field containing the name of the console to which this message is to be directed. It may be used as an alternative to, and under the same conditions as CONSID. The name must be left-justified and padded with blanks.

CART=address5 | (r5)
Specifies the address of an 8-byte field containing a command and response token to be associated with this message. When omitted for a non-connected message, a CART value of all 0s is assumed.

MF=I | L | (E,...)
Requests a macro expansion in-line (I) or in the list (L) or execute (E) format. Register notation may not be used for the list format. For the execute form, **address** specifies the address of a WTO parameter list generated by the list format. Registers 1 to 12 may be used for this address in register notation.

When the TEXT, CONSID, CONSNAME or CART operands are used, these keywords must also be specified in the list form, even if no values are assigned (for example, just CONSID=), to ensure that the right version of the parameter list is generated.

A successful WTO returns a 4-byte message ID in register 1, that can be used for subsequent WTOs (CONNECT parameter) or for the DOM macro.

When a message is issued for a partition with an active VSE/POWER ECHO option and with routing code 11 on and CONSID or CONSNAME omitted, routing code 11 is reset and the ECHO userid is inserted for CONSNAME. If no other routing code is on after resetting of routing code 11 and if the ECHO option is REPLY, the message is suppressed.

When a message with routing code 11 and CONSID/CONSNAME omitted is issued by the Attention Routine, routing code 11 is reset and the console ID of the command origin is inserted for CONSID.

When CONSID or CONSNAME were specified, or set as described above and no console with that name is active at the time the message was issued (nor a CMS user with that userid), return code 30 is presented (see also below). The message is processed anyway if routing codes other than 11 are on; it is ignored otherwise.

The CONNECT option is not supported for WTO issued from a VSE/ICCF interactive partition.

Return Codes in Register 15

00 Successful completion.

04 Number of lines passed was 0 (request is ignored), or message text length for a line was less than 1 (all lines up to error line were processed), or more than 10 lines were specified (only the first 10 lines were processed), or message text was truncated due to length longer than supported. Whenever the number of lines processed is smaller than specified, an ending line type is automatically generated. This also means that any attempt to connect a subsequent WTO to this message will fail.

08 Message ID specified by CONNECT operand was not found or terminates with a linetype of E or DE (request is ignored). This return code is also given when CONNECT is rejected for a non-privileged program.

0C Invalid line type (all lines up to the error line are processed). This return code is also given for non-privileged programs when a terminating line type was enforced by the system.

20 Processing was terminated due to shortage of message buffers and MCSFLAG option BUSYEXIT was specified.

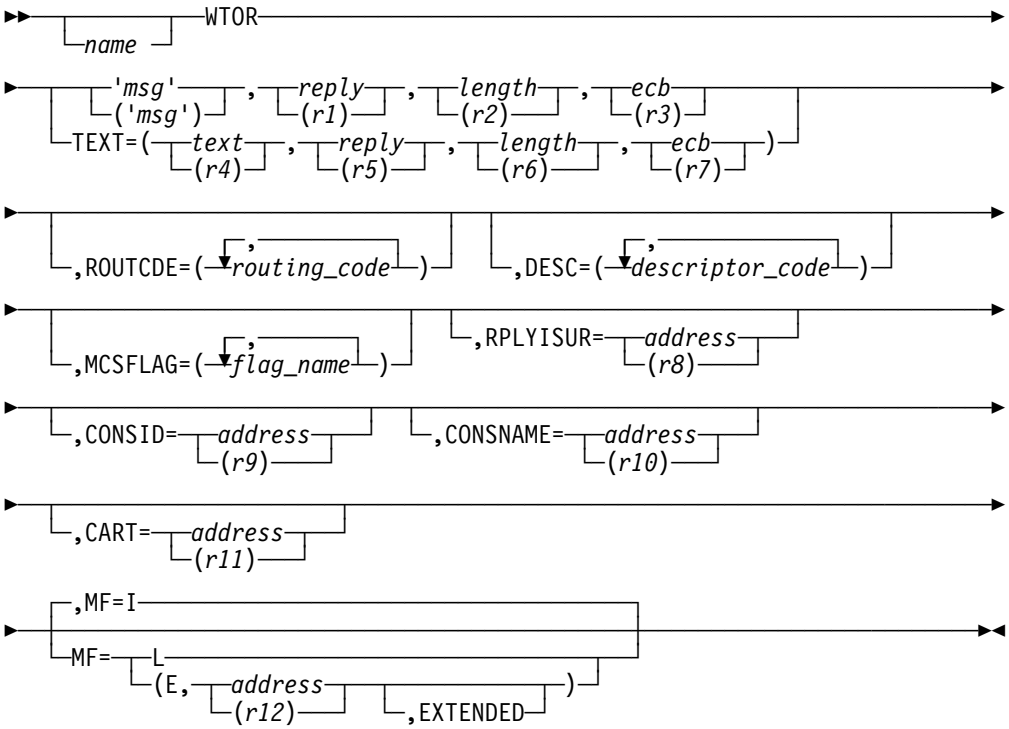
30 The console identified by CONSID or CONSNAME, or by an ECHO option in connection with routing code 11, is not active or not receiving (MSGDLVRY=NONE). If routing codes other than 11 were specified, the request was processed for those routing codes, otherwise it was ignored.

3C The system service responsible for WTO processing is not yet initialized. The request is ignored. This condition can only occur during the very early stage of system start-up.

Cancel Codes

21	One or more input parameters other than the special cases covered by return codes are invalid or not supported by VSE/ESA.
25	One or more of the specified addresses are invalid.
45	Mode violation (for example, caller is in AR-mode).

WTOR (Write to Operator with Reply) Macro



Except where noted otherwise, only registers 2 to 12 may be used for register notation.

Requirements for the caller:

AMODE: 24 or 31

RMODE: 24 or ANY

ASC Mode: Primary

The WTOR macro is used to issue a console message and to receive a reply. It allows to use a message text address (TEXT operand) in addition to literal text, to direct a message (typically a command response) to a specific console (CONSID or CONSNAME operand), to correlate command responses with the original command (CART operand), and to receive an identification of the console that entered the reply (RPLYISUR operand).

Both source and object level compatibility is preserved for programs using the WTOR version of earlier VSE/ESA releases. However, the VSE/ESA 2.1.0 default expansion is backward incompatible. If none of the 2.1.0-unique operands is used, a backward-compatible expansion, suitable for execution on all VSE/ESA releases, may also be obtained with the VSE/ESA 2.1.0 WTOR macro, by issuing SPLEVEL SET=n with n < 4 prior to WTOR invocation (see also “SPLEVEL (Set and Test Macro Level) Macro” on page 382).

'msg' | ('msg')
Denotes a single-line literal message. The message text must be enclosed in quotes and may be up to 122 characters long. Quotes to be included in the message text must be coded as double quotes. All characters with a

value smaller than X'40' (except for the DBCS control characters SI and SO) or equal X'FF' are replaced by blanks.

reply | (r1)
Specifies the address of an area for receiving the reply.

length | (r2)
Specifies the length of the reply area. Replies can be up to 119 characters, and are truncated at the specified reply length.

ecb | (r3)
Specifies the address of an ECB that is posted when the reply is available. The address must be on a halfword boundary.

TEXT=...
text denotes the address of an area containing a 2-byte length, followed by a character string of that length. The maximum length is 122 characters. All characters with a value smaller than X'40' (except for the DBCS control characters SI and SO) or equal X'FF' are replaced by blanks.

reply, **length**, and **ecb** have the same meaning as described above.

ROUTCDE=(routing code...)
Specifies one or more routing codes as decimal numbers in the range 1-128, separated by commas or by a hyphen to indicate a range.

Routing codes determine, along with the CONSID or CONSNAME operand (see below) and the VSE/POWER JECL JOB ECHO option, on which console(s) the message is to be displayed and whether or not the message is to be logged on the hardcopy file. The meaning associated with each code is the same as for WTO.

When WTOR is issued from an ICCF interactive partition and only routing code 11 is specified, the message is only displayed on the ICCF terminal and can only be replied from there. Such a message is not logged on the HC file and is not affected by DOM. If other routing codes are (also) present, the message is (also) routed and logged according to the general rules, but cannot be replied from the ICCF terminal.

DESC=(descriptor code...)
Specifies one or more descriptor codes as decimal numbers in the range 1-16. This operand has no effect for WTOR and is only provided for compatibility. A descriptor code of 7 is always assumed.

MCSFLAG=(keyword...)
Specifies one or more keywords requesting some special handling for this message. Only the following specifications are supported by VSE/ESA:

RESP The message is a command response.

HRDCPY The message is only to be logged on the hardcopy file. This option causes the WTOR ECB to be posted as soon as logging is completed, and may be used to synchronize job processing with logging. All routing parameters (ROUTCDE, CONSID, CONSNAME) are ignored.

RPLYISUR=address | (r8)
Specifies the address of a 12-byte area, where the system stores the 8-byte name and the 4-byte ID of the console where the reply was entered. Bit 0

being on in the console ID indicates to VSE/ESA a console without master authority.

CONSID=address | (r9)
Specifies the address of a 4-byte field or a register containing the ID of the console to which this message is to be directed.

CONSNAME=address | (r10)
Specifies the address of an 8-byte field containing the name of the console to which this message is to be directed, The name may be used as an alternative to the console ID and under the same conditions as CONSID. The name must be left justified and padded with blanks.

CART=address | (r11)
Specifies the address of an 8-byte field containing a command and response token to be associated with this message. When omitted for a non-connected message, a CART value of all zeros is assumed.

MF=I | L | (E,...)
Requests a macro expansion in-line (I) or in the list (L) or execute (E) format. Register notation may not be used for the list format.

When the TEXT, RPLYISUR, CONSID, CONSNAME or CART operands are used, an extended form of the parameter list is required. To ensure that the right version of the parameter list is generated, these keywords must also be specified in the list form, even if no values are assigned, for example, just CONSID= or TEXT=().

For the execute form, MF also specifies the address of a WTOR parameter list generated by the list format, and whether the extended form (EXTENDED) is used. Registers 1 to 12 may be used for the address of the parameter list in register notation.

A successful WTOR returns a 4-byte message ID in register 1, which can be used for a subsequent DOM macro.

When a message is issued for a partition with an active VSE/POWER JECL JOB ECHO option and routing code 11 on and CONSID or CONSNAME omitted, routing code 11 is reset and the ECHO userid is inserted for CONSNAME.

When a message with routing code 11 and CONSID/CONSNAME omitted is issued by the Attention Routine, routing code 11 is reset and the console ID of command origin is inserted for CONSID.

When CONSID or CONSNAME were specified, or set as described above, and no console with that name is active at the time the message was issued, nor a CMS user with that userid, return code 30 is given (see also below). The message is processed anyway, if routing codes other than 11 are on; otherwise it is ignored.

WTOR macro processing for an application program running in an VSE/ICCF pseudo partition: WTOR messages with routing code 11 and other routing codes set will be displayed as non-read messages on the VSE/ICCF terminal. The "read" goes to the other consoles (determined by the other routing codes) and a reply will only be accepted from these consoles.

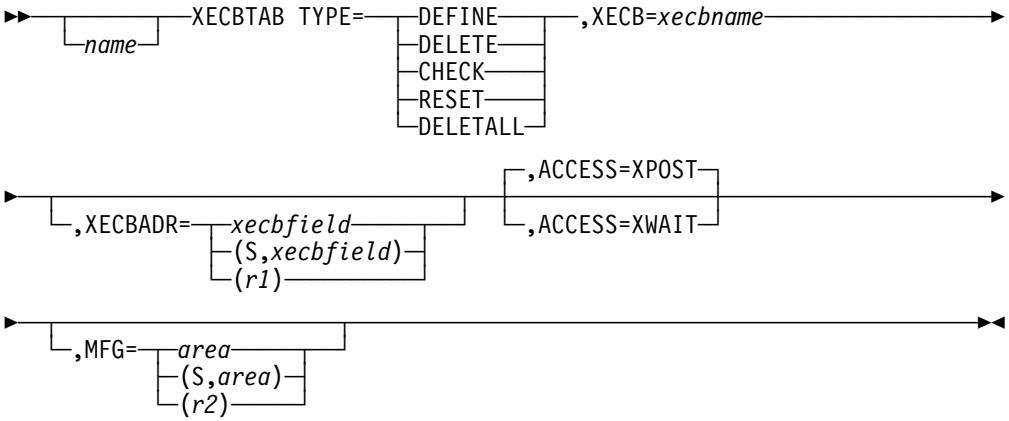
Return Codes in Register 15

00	Successful completion.
04	Number of lines passed was 0 (request is ignored), or message text length was less than 1 (request is ignored), or message text was truncated due to length longer than supported.
30	The console identified by CONSID or CONSNAME, or by an ECHO option in connection with routing code 11, is not active or not receiving (MSGDLVRY=NONE). If routing codes other than 11 were specified, the request was processed for those routing codes, otherwise it was ignored.
3C	The system service responsible for WTOR processing is not yet initialized. The request is ignored. This condition can only occur during the very early stage of system start-up.

Cancel Codes

21	One or more input parameters other than the special cases covered by return codes are invalid or not supported by VSE/ESA.
25	One or more of the specified addresses are invalid.
45	Mode violation (for example, caller is in AR-mode).

XECBTAB (Cross-Partition Event Control Block Table) Macro



Requirements for the caller:

AMODE: 24

RMODE: 24

ASC Mode: Primary

The macro can be used to:

- Define, for the specified cross-partition event control block (XECB), an entry in the supervisor's XECB table.
- Delete an XECB table entry.
- Check for the presence of an XECB table entry.
- Reset an XECB table entry.

An XECB for which an entry has been defined to the supervisor can be referred to by XPOST and XWAIT macros. An XECB can be referred to also by a WAIT or WAITM macro if the task issuing the macro has previously defined the XECB with ACCESS=XWAIT.

The XECBTAB (and the XPOST and XWAIT) macro can be used to establish a connection between two private static partitions in the same address space. XECBTAB does not work in dynamic partitions; use the XPCC macro instead.

TYPE=CHECK | DEFINE | DELETE | DELETALL | RESET
The operand specifies the type of operation to be performed:

TYPE=CHECK
Causes the system to check whether or not an entry for a specific XECB has been defined already. If that entry exists, the system returns the address of both the XECB and the associated XECB table entry. For more information, turn to Figure 52 on page 416.

TYPE=DEFINE
Causes a new XECB table entry to be defined to the supervisor.

TYPE=DELETE
Causes an entry to be deleted from the supervisor's XECB table. TYPE=DELETE can be specified only for an XECB for which an entry has been defined previously in the same program.

TYPE=DELETALL
This specification causes three actions. It:

- Deletes all entries in the XECB table that were defined previously by the issuing task.
- Breaks the communication between any XECB owner and the issuing task (that is, clears the information in the XECB table that indicates that the issuing task communicates with the XECB owner).
- Posts as ready-to-run all tasks that are waiting for an XPOST by the issuing task. Also, it sets on the abnormal termination bit in the ECB (bit 1 of byte 2). A task waiting because of an XWAIT on the XECB gets a return code of X'08' if this task owns the XECB.

Notes:

1. If DELETALL is specified, the XECB and ACCESS operands must not be specified. XECBTAB with TYPE=DELETALL specified does not provide a return code; all registers remain unchanged.
2. XPOST partition abnormal termination is not indicated to tasks using WAIT or WAITM macro.

TYPE=RESET
Causes the system to clear the information in the supervisor XECB table that indicates which task communicates with the program having defined the XECB. RESET can be specified only for an XECB for which an entry has been previously defined in the same program.

After RESET, any task can attempt to establish a new connection with the owner. With ACCESS=XPOST, however, if the task currently connected to the XECB is issuing an XWAIT macro (at the time of RESET), this task probably establishes connection again, nullifying the RESET operation.

XECB=xecbname
Specifies the name of the XECB. If the XECBADR operand is not present, xecbname is the symbolic address of the four-byte (or larger) XECB field. If, however, XECBADR is specified, xecbname is the name by which the control block is known between partitions. The symbolic address of the control block field is given by XECBADR.

The XECB field must be defined in your program, except when TYPE=CHECK is specified: in that case, the XECB field may be defined in another program.

XECBADR=xecbfield | (S,xecbfield) | (r1)
XECBADR is used only if TYPE=DEFINE is specified. It provides the symbolic address of the four-byte (or larger) field that is to be used as XECB.

ACCESS=XPOST | XWAIT

This operand can be used together with TYPE=DEFINE to specify that the program is allowed to post the XECB or wait for another program to do the posting.

XPOST, the default, specifies that the program is allowed to post the XECB. Specifying XPOST implies that only one other active task is allowed to issue an XWAIT macro against the XECB.

XWAIT specifies that the program will be allowed to wait for one other task to post the XECB.

MFG=area | (S,area) | (r2)

The operand is required if the program that issues the XECBTAB macro is to be reenterable. The operand specifies the address of a 64-byte storage area, that is, storage which your program obtains by a GETVIS macro. This area is needed for use by the system during execution of the macro.

The MFG operand is useful only together with XECBADR coded in either of these two notations: (S,xecbfield) or (r1).

Feedback Information

Figure 52 shows the return codes that are supplied in register 15. The illustration also indicates whether or not the system returns the addresses of the specified XECB and the associated table entry in registers 1 and 14, respectively.

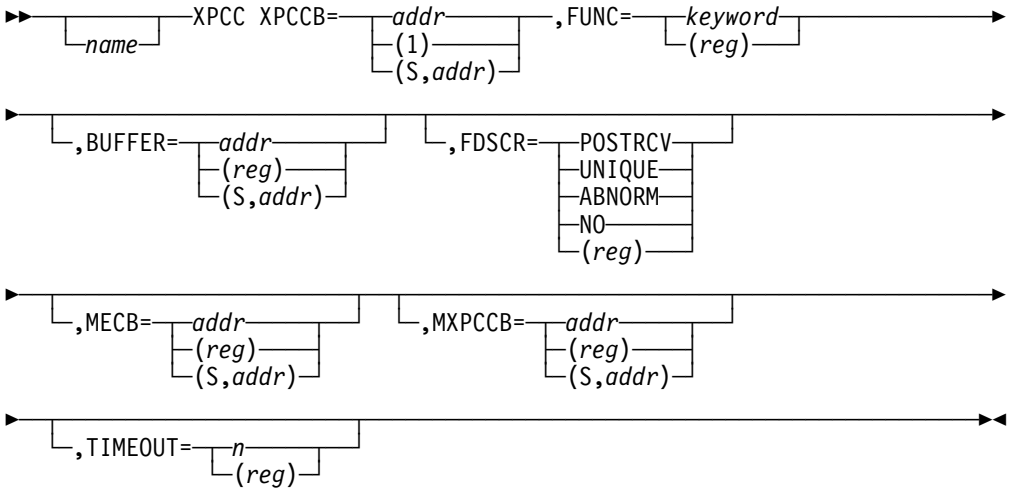
Figure 52. XECBTAB Feedback Information

TYPE=	X'00'	X'04'	X'08'
CHECK	The named XECB was found in the table (see also Note 1).	The named XECB was not found (see also Note 2).	The named XECB was found in the table (see also Note 3).
DEFINE	The named XECB is stored in the table (see also Note 1).	The named XECB is already in the table (see also Note 2).	The named XECB is full (see also Note 2).
DELETE	The named XECB is removed from the table (see also Note 2).	The named XECB was not found (see also Note 2).	The issuing program did not define the XECB (see also Note 2).
RESET	Communication bytes of the named XECB were cleared (see also Note 2).	The named XECB was not found (see also Note 2).	The issuing program did not define the XECB (see also Note 2).

Notes:

1. Register 1 contains the address of the XECB and register 14 the address of the table entry.
2. Registers 1 and 14 are set to zero.
3. The issuing program and the program having defined the XECB do not run in private partitions within the same address space (registers 1 and 14 are set to 0). A program in a private static partition can continue to use an XECB to communicate with VSE/POWER. However, since the VSE/POWER XECBTAB-based macros GETSPOOL, PUTSPOOL, and CTLSPOOL do not work for dynamic partitions, use the XPCC macro instead, together with the PWRSPL macro.

XPCC (Cross-Partition Communication) Macro



Requirements for the caller:

AMODE: 24 or 31

RMODE: 24 or ANY

ASC Mode: Primary

The macro invokes the cross-partition communication service, which allows communication between two application programs (two different VSE tasks). For the values and meanings of the return and reason code information related to the various XPCC requests, see the “MAPXPCCB (Map Cross-Partition Control Block) Macro” on page 313.

XPCCB=addr | (1) | (S,addr)

Defines the address of the XPCCB, which is the control block that contains all request-related information. The control block is set up with the XPCCB macro. Depending on the request, only certain fields of the control block are used (for details, refer to the description of the various functions under “Cross-Partition Communication” in the *VSE/ESA System Macro User's Guide* .)

The address of the XPCCB is treated as a 3-byte address if the issuer of the macro is operating in 24-bit mode, and as a 4-byte address if operating in 31-bit mode.

FUNC=keyword | (reg)

Defines the specific function to be requested from the XPCC service. Depending on the type of request, an application may use the following keywords:

1. Initialization request:

   IDENT Identify an application to the XPCC service.

2. Connection-related requests:

   CONNECT Connect an application to another application. DISCONN Terminate a connected link to another application (if no data transmission is going on at the moment).

   DISCPRG Terminate a connection unconditionally. This may interrupt the transfer of data.

   DISCALL Disconnect unconditionally all connections for a certain application.

3. Data transmission requests:

   SEND Send data to another application.

   SENDR Send data and request a reply back from the receiver.

   SENDI Send data into a predefined area and give SENDI-state to partner.

   RECEIVE Receive data.

   REPLY Send a reply back to the sender.

   CLEAR Revoke a previously initiated SEND request from the connection (used by the sender).

   PURGE The receiver purges the data, because he is not able to receive it.

4. Termination requests:

   TERMIN Terminate XPCC usage (if all links are already disconnected).

   TERMPRG Terminate unconditionally. This may interrupt the transfer of data.

   TERMQSCE No termination yet, but new connections to this application are not granted anymore.

Depending on the selected function, IJBXFCT of the cross-partition control block (XPCCB) is set accordingly.

If the format **FUNC=(reg)** is used, the specified register must have been loaded with the corresponding function byte value. These values can be found in the program listing under label IJBXFCT in the mapping macro MAPXPCCB (see Figure 46 on page 316).

BUFFER=addr | (reg) | (S,addr)

This operand may optionally be used in connection with SEND, SENDR, SENDI, RECEIVE, and REPLY requests to dynamically provide a data area from where (SEND, SENDR, SENDI, REPLY) or to which (RECEIVE) data is to be moved. If you use this operand, the corresponding BUFFER field in the XPCCB control block is overwritten.

If the address is loaded into a register, it must be a 4-byte address.

For a RECEIVE or REPLY request, the BUFFER address must point to an 8-byte area with the following format:

Bytes	Description
0,bit 0 0 – 3 4 – 7	ON Data area address (31-bit address) Length of data area

For a SEND or SENDR request, the BUFFER address must point to an address list as shown below, where each entry has the following format:

Bytes	Description
0,bit 0 0 – 3 4 – 7	Indicator bit: OFF : Not last entry in list ON : Last entry in list Data address of buffer segment (31-bit) Length of buffer segment

You can specify up to 256 entries of this format in one buffer address list. The buffer segments are concatenated and passed as one buffer.

Note: For performance reasons, it is recommended that buffers should start at a page boundary or, if smaller than a page, do not cross the page boundary.

FDSCR=POSTRCV | UNIQUE | ABNORM | NO | (reg)

This operand specifies an option for the requested function. If omitted, the current value of the function descriptor field IJBXFDSC of the XPCCB is used (see also Figure 47 on page 317).

POSTRCV Can be specified with the SENDR function. The sender is notified when the data has been received by the other side. IJBXCECB is posted and the reason code field IJBXREAS is set to IJBXRECX. If register notation is used, the register must be loaded with IJBXPOST.

UNIQUE Can be specified with the IDENT function. It ensures that the application name specified in the XPCCB macro is unique in this VSE system. If the application name is already known in the system, the request is rejected with return code X'08' in register 15 and return code field IJBXRETC set to IJBXDUP. If the IDENT request is granted, each subsequent request with the same application name is rejected with the same return information. If register notation is used, the register must be loaded with IJBXUNIQ.

ABNORM Can be specified with the DISCONN and DISCPRG functions. The other side is posted an abnormal-end condition of the current task, with reason code field IJBXREAS set to IJBXABDC. If register notation is used, the register must be loaded with IJBXFDAB.

NO Can be specified with any function. It forces the function descriptor code to X'00'.

If register notation is used, the register must be loaded with binary zeros.

MECB=addr | (reg) | (S,addr)

This operand may optionally be used with the CONNECT function. It specifies the address of an ECB within the user's partition. This 'main ECB' is always posted when any of the XPCC ECBs is posted. All tasks waiting

on the MECB are set to 'ready-to-run'. The application owning the main ECB is responsible for resetting the traffic bit in the main ECB.

If the address is loaded into a register, it must be a 4-byte address.

MXPCCB=addr | (reg) | (S,addr)
This operand may optionally be used with the CONNECT function. It defines the XPCCB which contains the identify token to be used as input in the CONNECT request.

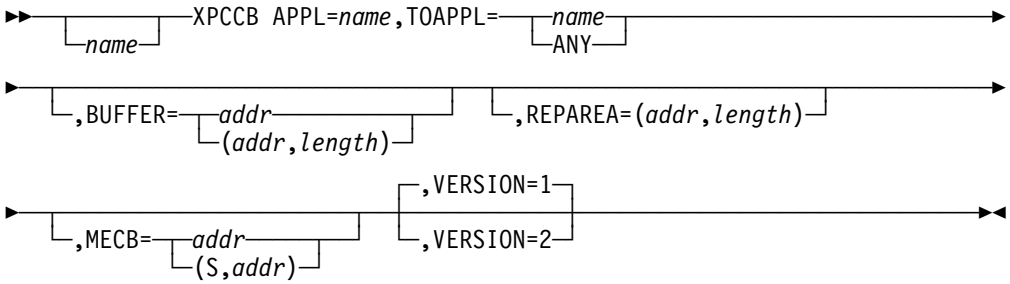
If the address is loaded into a register, it must be a 4-byte address.

TIMEOUT=n | (reg)
Specifies the number of seconds the issuer of a CONNECT request is prepared to wait. The value for **n** must be in the range of 0 to 255. After the specified time interval is exhausted, IJBXCECB, IJBXSECB, and IJBXRECB are posted and reason code IJBXTOUT is set in IJBXREAS. The XPCC user has to respond with either an XPCC FUNC=DISCONN or DISCPRG request.

In register notation, byte 3 of the register contains the value n. 0 means an immediate request. In this case the CONNECT request is rejected with IJBXTIMO if the partner has not already issued CONNECT.

If the TIMEOUT operand is specified, the XPCCB must be defined with VERSION=2.

XPCCB (Cross-Partition Control Block) Macro



Required RMODE: **24** or **ANY**

This macro sets up a cross-partition communication control block XPCCB. Each connection or communication path between the two applications is represented by a unique XPCCB. The address of the XPCCB is indicated in the XPCC macro. The corresponding DSECT is generated by means of the MAPXPCCB macro. This mapping DSECT may be used to reference or modify the control block fields at execution time when setting up a VSE XPCC request or when checking the status of the XPCC connection.

APPL=name
Specifies the name of the application requesting XPCC service. It may be up to eight bytes long and must not contain blanks or all binary zeros (printable characters are recommended). Application names starting with SYS are reserved for programs written by IBM.

TOAPPL=name | ANY
Specifies the name of the application to which communication is to be established.

ANY indicates that an open communication link is to be set up, which means that any other application can link up. (ANY must not be used as application name.)

BUFFER=addr | (addr,length)
Specifies the address of one or more buffer areas for the data transmission request, where **addr** is a pointer to a list of 8-byte fields, as described under the BUFFER operand in the “MAPXPCCB (Map Cross-Partition Control Block) Macro” on page 313. If the list contains only one entry, it can be specified directly by **(addr,length)** instead of **addr**.

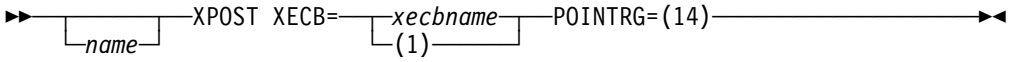
REPAREA=(addr,length)
Specifies, for the SENDR function, the address of the data area into which the reply is to be moved.

MECB=addr | (S,addr)
Specifies, for the CONNECT request, the address of a 'main ECB', which is always posted when any of the XPCC ECBs is posted. All tasks waiting for the MECB are taken out of the wait state.

VERSION=1 | 2

Indicates the version of the XPCCB. Two versions of the XPCCB are maintained: VERSION=1 is the default. VERSION=2 is required if the application wants to make use of the TIMEOUT, SENDI or MECB functions of the XPCC support.

XPOST (Cross-Partition Post) Macro



Requirements for the caller:

AMODE: 24

RMODE: 24

ASC Mode: Primary

The macro provides for cross-partition communication by posting the specified XECB (the macro sets bit 0 of byte 2 to 1).

An XPOST macro issued against an XECB causes the task waiting for this XECB to be removed from the wait state. This task may have issued an XWAIT, a WAIT or a WAITM with a previously defined XECB. The task may have been activated in the same or in another private partition within the same address space.

If the XPOST macro is used in a mainline loop, the macro should be preceded by a test which ensures that the other partition's task waiting for the event that is being posted must receive control and execute the function for which this event is a prerequisite.

To perform this test, a second XECB needs to be defined. This XECB allows the originally waiting task in its mainline loop to post completion of its function as an event for which the originally posting task must wait.

Resetting bit 0 of byte 2 of the XECB is a user responsibility.

Once a task has issued an XPOST macro for an XECB (with ACCESS=XWAIT), no other task can issue an XPOST for this XECB, until the connection is ended.

XECB=xecbname | (1)
Specifies the name of the XECB to be posted. The name you specify must be the same as the one used to define the XECB. If register notation is used, the specified register must point to an 8-byte character field that contains the XECB name left-justified and padded with blanks. Do not specify 14 or 15 if you choose to use ordinary register notation.

POINTRG=(14)
Specifies the register that points to the XECB table entry associated with the named XECB. Do not specify register 1 or 15 if you choose to use ordinary register notation.

To obtain the address of the associated XECB table entry, issue earlier in the program an XECBTAB macro for the same XECB and with TYPE=CHECK or TYPE=DEFINE specified. When the system executes the XECBTAB macro, it returns, in register 14, the address of the pertinent XECB table entry. Figure 53 on page 426, which shows a coding example for the use of the XWAIT macro, applies to the XPOST macro accordingly.

If the POINTRG register contains 0 (or any invalid value), all entries in the XECB table are searched to determine the correct address; no error is indicated.

Return Codes in Register 15

When the system returns control to the issuing task, register 15 contains one of the following return codes:

00 Successful completion. The named XECB has been posted.

04 The named XECB has no associated table entry in the XECB table.

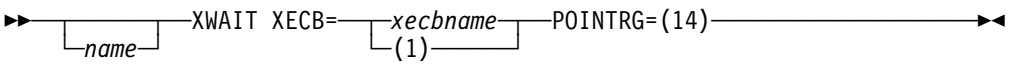
0C The named XECB is not within the current address space.

0D The XECB referred to in the XPOST macro was defined with ACCESS=XPOST specified in the XECBTAB macro, but the task that issued the XPOST macro does not own this XECB.

0E The XECB referred to in the XPOST macro was defined with ACCESS=XWAIT specified in the XECBTAB macro and either (1) the task that issued the XPOST macro also defined the XECB or (2) the XECB has been posted previously during the same execution by another task.

Note: Following the execution of an XPOST macro, registers 1 and 14 are set to zero.

XWAIT (Cross-Partition Wait) Macro



Requirements for the caller:

AMODE: 24

RMODE: 24

ASC Mode: Primary

The macro enables the issuing task to wait for an XECB to be posted by another task. This other task may be executing in the same or in another private partition within the same address space. Control returns to the issuing task when the XECB is posted or if an error condition is detected.

Once a task has issued an XWAIT macro for an XECB (with ACCESS=XPOST) to be posted, no other task can issue an XWAIT for this XECB, until the connection is ended.

XECB=xecbname | (1)

Specifies the name of the XECB, which may be defined in the same or another program. The name you specify must be the same as the one used to define the XECB. If register notation is used, the specified register must point to an eight-byte field that contains the name of the XECB left-justified and padded with blanks. Do not specify register 14 or 15 if you choose to use ordinary register notation.

POINTRG=(14)

Specifies the register that points to the XECB table entry associated with the named XECB. Do not specify register 1 or 15 if you choose to use ordinary register notation.

To obtain the address of the associated XECB table entry, issue earlier in the program an XECBTAB macro for the same XECB and with the TYPE=CHECK or TYPE=DEFINE specified. When the system executes the XECBTAB macro, it returns, in register 14, the address of the pertinent XECB table entry. Figure 53 on page 426 shows a coding example for the use of the XWAIT macro; in that example, the required continuation character is not shown. The example assumes that the XECB was defined by a program running in another partition.

Return Codes in Register 15

When the system returns control to the issuing task, register 15 contains one of the following return codes:

00 Successful completion. The named XECB has been posted.

04 The named XECB has no associated table entry in the XECB table or the owner of the XECB issued a DELETALL.

08 The other task using this XECB has broken communication without issuing an XPOST. The task issuing the XWAIT is owner of the XECB.

0C The named XECB is not within the current address space.

0D The XECB referred to in the XWAIT macro was defined with ACCESS=XWAIT specified in the XECBTAB macro, but the task that issued the XWAIT macro does not own this XECB.

0E The XECB referred to in the XWAIT macro was defined with ACCESS=XPOST specified in the XECBTAB macro and either (1) the task that issued the XWAIT macro also defined the XECB or (2) the task did not define the XECB, but another task is already waiting for the XECB to be posted.

Note: Following the execution of an XWAIT macro, registers 1 and 14 are set to zero.

Defining the XECB in a program running in another partition

```
             ...
            XECBTAB   TYPE=DEFINE,
                      XECB=MYECB
             ...
    MYECB   DC        F'0'
             ...
```

The use of the XWAIT macro:

```
             ... ... ...
    WAITLP  XECBTAB  TYPE=CHECK,
                     XECB=MYECB
            LTR      15,15
            BNZ      ERROR
            LA       1,XECBNAME
            XWAIT    XECB=(1),
                     POINTRG=(14)
             ... ... ...
    XECBNAME DC      CL8'MYECB   '
```

If the register specified with POINTRG contains 0 (or any invalid value), then all XECBs are searched to determine the correct address. No error is indicated.

Figure 53. Coding Example Showing the Use of XECBTAB with TYPE=CHECK and XWAIT

YEAR224 Macro

This macro can be used to complement a 2-digit year field, supplied as input, with a 2-digit century field, depending on a specified 100 year window relative to the current year. The format is as follows:

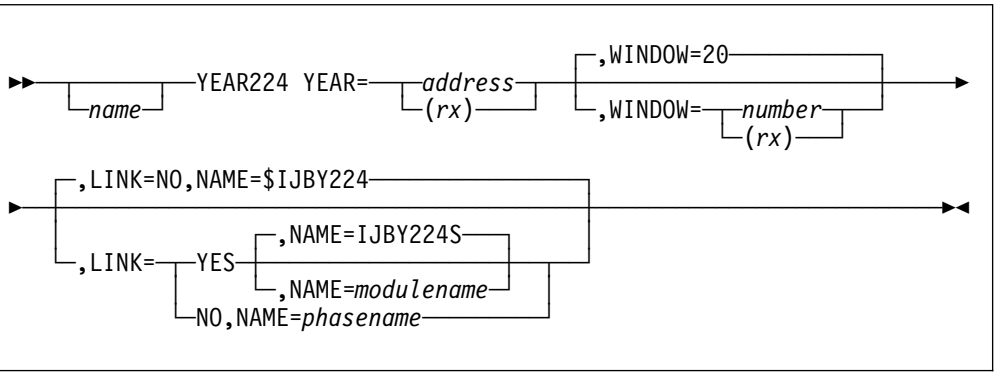


Figure 54. Syntax of YEAR224 Macro

Requirements for the caller:

AMODE: 24 or 31
RMODE: ANY
ASC Mode: Primary

Register usage convention:

(rx) For register notation (YEAR and WINDOW parameter) any general register from 2 to 12 can be used.

Register 0 Is used for input to pass the WINDOW parameter and returns one of the following reason codes:

- **0** No errors (register 15 also contains 0).
- **1** YEAR input is not numeric.
- **2** WINDOW is invalid (negative or larger than 99).

Register 1 Is used for input to pass the YEAR parameter.

Register 13 Is assumed to contain the address of a 72-byte save area.

Register 14 Is used as link register.

Register 15 Is used at input for the address of the service routine, and returns one of the following return codes:

- **0** No errors.
- **8** Input is invalid, as indicated by the reason code in register 0.

Input Parameters:

YEAR=address|(rx)
Specifies the address of a 4-byte area containing the 2-digit year input in the form "..yy". The content of substring ".." is ignored on input and is replaced by the century on output, resulting in a 4-digit year "yyyy". For register notation (rx), register 2 through 12 can be used.

WINDOW=number|20|(rx)
Specifies a number between 0 and 99, that is interpreted as the forward width of a 100 year window relative to the current year. The default is 20. For register notation (rx), register 2 through 12 can be used.

LINK=YES|NO Determines how the service routine is invoked. With the default LINK=NO, the macro invokes, by default, the SVA phase $IJBY224 (no link-editing is required). With LINK=YES, the macro invokes IJBY224S as the default CSECT for link-editing. In both cases, you can also specify your own phase or CSECT name.

LINK=YES means that the call sequence generates an external reference to the entry point of the service routine (CSECT) causing it to be linked with the calling program.

NAME=name|$IJBY224|IJBY224S
Specifies a user-defined phase or CSECT as the service routine to be invoked. If no name is specified, the following default names are used, as mentioned in the LINK parameter description:

- For LINK=NO, phase name $IJBY224.
- For LINK=YES, CSECT name IJBY224S.

The macro expansion and service routine are both reentrant. The service routine uses the GETIME macro and references the COMREG field SYSDATE to obtain the current date, and is therefore VSE-dependent. It may also be used in a CMS environment under VM/ESA.

Example:

```
            YEAR224 YEAR=yyyy
            ...
   yyyy     DC    '..02'
```

In the above example, the macro complements field yyyy to 2002, since 2002 is the year ending with 02 in the default window (1996-79, 1996+20) = (1917, 2016). Refer to *VSE/ESA Planning* for further details about the default window used.

Appendix A. Control Character Codes

CTLCHR=ASA Option

If the ASA option is chosen, a control character must appear in each record. An invalid control character for a printer causes the system to issue a message and to cancel the job.

If the control character for a card device is not V or W, the card is selected into stacker 1.

The codes are listed below:

Code	Interpretation
blank	Space one line before printing (see Note below)
0	Space two lines before printing
-	Space three lines before printing
+	Suppress space before printing
1	Skip to channel 1 before printing (see Note below)
2	Skip to channel 2 before printing
3	Skip to channel 3 before printing
4	Skip to channel 4 before printing
5	Skip to channel 5 before printing
6	Skip to channel 6 before printing
7	Skip to channel 7 before printing
8	Skip to channel 8 before printing
9	Skip to channel 9 before printing
A	Skip to channel 10 before printing
B	Skip to channel 11 before printing
C	Skip to channel 12 before printing
V	Select stacker 1
W	Select stacker 2
X	Select stacker 3
Y	Select stacker 4
X	Select stacker 5
5A	CPDS (compose page data stream) record, accepted for DTFPR

Note: For a print (not associated) file on an IBM 3525, either a space one line or a skip to channel 1 must be used to print on the first line of a card. For a print associated file, only space one line must be used to print on the first line of a card.

© Copyright IBM Corp. 1990, 1999

CTLCHR=YES Option

The control character for this option is the command-code portion of the CCW used in printing a line or spacing the forms. The control codes are listed in below:

Stacker Selection Codes

Hex Code	Card-Code	Function
Stacker Selection on a Card Input or Output Device (except IBM 1442):		
01	12-1-9	Select into stacker 1
41	12-0-1-9	Select into stacker 2
Stacker Selection Specific for an IBM 2540:		
81	12-0-1	Select into stacker 3 (see Note below)
Stacker Selection on a Multifunction Card Machine:		
13	11-3-9	Primary hopper: select into stacker 1
23	0-3-9	Primary hopper: select into stacker 2
33	3-9	Primary hopper: select into stacker 3
43	12-0-3-9	Primary hopper: select into stacker 4
93	12-11-3	Secondary hopper: select into stacker 1
A3	11-0-3	Secondary hopper: select into stacker 2
B3	12-11-0-3	Secondary hopper: select into stacker 3
C3	12-3	Secondary hopper: select into stacker 4

Note: The code cannot be used if you defined your file with DTFDI.

Printer Control Codes

Hex Code	Card-Code	Function
Printer Control (Except for IBM 3525):		
01	12-1-9	Write (without automatic space)
09	12-1-8-9	Write and space 1 line
11	11-1-9	Write and space 2 lines
19	11-1-8-9	Write and space 3 lines
89	12-0-9	Write and skip to channel 1
91	12-11-1	Write and skip to channel 2
99	12-11-9	Write and skip to channel 3
A1	11-0-1	Write and skip to channel 4
A9	11-0-9	Write and skip to channel 5
B1	12-11-0-1	Write and skip to channel 6
B9	12-11-0-9	Write and skip to channel 7
C1	12-1	Write and skip to channel 8
C9	12-9	Write and skip to channel 9
D1	11-1	Write and skip to channel 10
D9	11-9	Write and skip to channel 11
E1	11-0-1-9	Write and skip to channel 12
0B	12-3-8-9	Space 1 line immediately
13	11-3-9	Space 2 lines immediately
1B	11-3-8-9	Space 3 lines immediately
8B	12-0-3-8	Skip to channel 1 immediately
93	12-11-3	Skip to channel 2 immediately
9B	12-11-3-8	Skip to channel 3 immediately
A3	11-0-3	Skip to channel 4 immediately
AB	11-0-3-8	Skip to channel 5 immediately
B3	12-11-0-3	Skip to channel 6 immediately
BB	12-11-0-3-8	Skip to channel 7 immediately
C3	12-3	Skip to channel 8 immediately
CB	12-0-3-8-9	Skip to channel 9 immediately
D3	11-3	Skip to channel 10 immediately
DB	12-11-3-8-9	Skip to channel 11 immediately
E3	0-3	Skip to channel 12 immediately
5A	11-2-8	CPDS (compose page data stream) record
03	12-3-9	No operation

Hex Code	Card-Code	Function
Printer Control for IBM 3525 (with Print Feature):		
0D	12-5-8-9	Print on line 1
15	11-5-9	Print on line 2
1D	11-5-8-9	Print on line 3
25	0-5-9	Print on line 4
2D	0-5-8-9	Print on line 5
35	5-9	Print on line 6
3D	5-8-9	Print on line 7
45	12-0-5-9	Print on line 8
4D	12-5-8	Print on line 9
55	12-11-5-9	Print on line 10
5D	11-5-8	Print on line 11
65	11-0-5-9	Print on line 12
6D	0-5-8	Print on line 13
75	12-11-0-5-9	Print on line 14
7D	5-8	Print on line 15
85	12-0-5	Print on line 16
8D	12-0-5-8	Print on line 17
95	12-11-5	Print on line 18
9D	12-11-5-8	Print on line 19
A5	11-0-5	Print on line 20
AD	11-0-5-8	Print on line 21
B5	12-11-0-5	Print on line 22
BD	12-11-0-5-8	Print on line 23
C5	12-5	Print on line 24
CD	12-0-5-8-9	Print on line 25

Appendix B. American National Standard Code for Information Interchange

American National Standard Code for Information Interchange (ASCII)

In addition to the EBCDIC mode, your VSE system accepts magnetic tape files written in ASCII, a 128-character 7-bit code. The high-order bit in this 8-bit environment is zero. ASCII is based on the specifications of the American National Standards Institute, Inc.

VSE processes ASCII files in EBCDIC with the help of two translate tables, which reside in the SVA. Using these tables, logical IOCS translates from ASCII to EBCDIC all data as it is read into the I/O area. For ASCII output, logical IOCS translates data from EBCDIC to ASCII just before writing the record.

Figure 55 shows the relative bit positions of the ASCII character set. An ASCII character is described by its column/row position in the table. The columns across the top of the figure list the three high-order bits. The rows along the left side of the figure are the four low-order bits.

For example, the letter P in ASCII is under column 5 and row 0 and is described in ASCII notation as 5/0. ASCII 5/0 and EBCDIC X'50' represent the same binary configuration (B'01010000'). However, P graphically represents this configuration in ASCII and & in EBCDIC. ASCII notation is always expressed in decimal. For example, the ASCII Z is expressed as 5/10 (not 5/A).

For those EBCDIC characters that have no direct equivalent in ASCII, the substitute character (SUB) is provided during translation. See Figure 56 on page 434 for ASCII to EBCDIC correspondence.

Note: If an EBCDIC file is translated into ASCII, and you translate back into EBCDIC, this substitute character may not receive the expected value.

© Copyright IBM Corp. 1990, 1999

ASCII Code

Figure 55. ASCII Character Set

b_7 b_6 b_5						0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
Bits[3]	b_4	b_3	b_2	b_1	Row ▾ Column →	0	1	2	3	4	5	6	7
	0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
	0	0	0	1	1	SOH	DC1	¦[1]	1	A	Q	a	q
	0	0	1	0	2	STX	DC2	"	2	B	R	b	r
	0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
	0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
	0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
	0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
	0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
	1	0	0	0	8	BS	CAN	(	8	H	X	h	x
	1	0	0	1	9	HT	EM	)	9	I	Y	i	y
	1	0	1	0	10	LF	SUB	*	:	J	Z	j	z
	1	0	1	1	11	VT	ESC	+	;	K	[	k	{
	1	1	0	0	12	FF	FS	,	<	L	\	l	¦
	1	1	0	1	13	CR	GS	-	=	M	]	m	}
	1	1	1	0	14	SO	RS	.	>	N	^[2]	n	~
	1	1	1	1	15	SI	US	/	?	O	_	o	DEL

1 The graphic | (Logical OR) may also be used instead of ! (Exclamation Point).
2 The graphic ¬ (Logical NOT) may also be used instead of ^ (Circumflex).
3 The 7 bit ASCII code expands to 8 bits when in storage by adding a high order 0 bit.
EXAMPLE: Number sign (#) is represented by b_7 b_6 b_5 b_4 b_3 b_2 b_1
0 0 1 0 0 0 1 1

Control Character Representations

NUL	Null	DLE	Data Link Escape (CC)
SOH	Start of Heading (CC)	DC1	Device Control 1
STX	Start of Text (CC)	DC2	Device Control 2
ETX	End of Transmission (CC)	DC3	Device Control 3
EOT	End of Transmission (CC)	DC4	Device Control 4
ENQ	Enquiry (CC)	NAK	Negative Acknowledge (CC)
ACK	Acknowledge (CC)	SYN	Synchronous Idle (CC)
BEL	Bell	ETB	End of Transmission Block (CC)
BS	Backspace (FE)	CAN	Cancel
HT	Horizontal Tabulation (FE)	EM	End of Medium
LF	Line Feed (FE)	SUB	Substitute
VT	Vertical Tabulation (FE)	ESC	Escape
FF	Form Feed (FE)	FS	File Separator (IS)
CR	Carriage Return (FE)	GS	Group Separator (IS)
SO	Shift Out	RS	Record Separator (IS)
SI	Shift In	US	Unit Separator (IS)
		DEL	Delete

(CC) Communication Control
(FE) Format Effector
(IS) Information Separator

Special Graphic Character

SP	Space	<	Less Than
!	Exclamation Point	=	Equals
|	Logical OR	>	Greater Than
"	Quotation Marks	?	Question Mark
#	Number Sign	@	Commercial At
$	Dollar Sign	[	Opening Bracket
%	Percent	\	Reverse Slant
&	Ampersand	]	Closing Bracket
'	Apostrophe	^	Circumflex
(	Opening Parenthesis	¬	Logical NOT
)	Closing Parenthesis	_	Underline
*	Asterisk	`	Grave Accent
+	Plus	{	Opening Brace
,	Comma	¦	Vertical Line
-	Hyphen (Minus)		(This graphic is
.	Period (Decimal Point)		stylized to
/	Slant		distinguish it
:	Colon		from Logical OR)
;	Semicolon	}	Closing Brace
		~	Tilde

Figure 56 (Page 1 of 3). ASCII to EBCDIC Correspondence

	ASCII				EBCDICI				
Character	Col	Row	Bit Pattern		Col	Row	Bit Pattern		Comments
					(in Hex)				
NUL	0	0	0000	0000	0	0	0000	0000	
SOH	0	1	0000	0001	0	1	0000	0001	
STX	0	2	0000	0010	0	2	0000	0010	
ETX	0	3	0000	0011	0	3	0000	0011	
EOT	0	4	0000	0100	3	7	0011	0111	
ENQ	0	5	0000	0101	2	D	0010	1101	
ACK	0	6	0000	0110	2	E	0010	1110	
BEL	0	7	0000	0111	2	F	0010	1111	
BS	0	8	0000	1000	1	6	0001	0110	
HT	0	9	0000	1001	0	5	0000	0101	
LF	0	10	0000	1010	2	5	0010	0101	
VT	0	11	0000	1011	0	B	0000	1011	
FF	0	12	0000	1100	0	C	0000	1100	
CR	0	13	0000	1101	0	D	0000	1101	
SO	0	14	0000	1110	0	E	0000	1110	
SI	0	15	0000	1111	0	F	0000	1111	
DLE	1	0	0001	0000	1	0	0001	0000	
DC1	1	1	0001	0001	1	1	0001	0001	
DC2	1	2	0001	0010	1	2	0001	0010	
DC3	1	3	0001	0011	1	3	0001	0011	
DC4	1	4	0001	0100	3	C	0011	1100	
NAK	1	5	0001	0101	3	D	0011	1101	
SYN	1	6	0001	0110	3	2	0011	0010	
ETB	1	7	0001	0111	2	6	0010	0110	
CAN	1	8	0001	1000	1	8	0001	1000	
EM	1	9	0001	1001	1	9	0001	1001	
SUB	1	10	0001	1010	3	F	0011	1111	
ESC	1	11	0001	1011	2	7	0010	0111	
FS	1	12	0001	1100	1	C	0001	1100	
GS	1	13	0001	1101	1	D	0001	1101	
RS	1	14	0001	1110	1	E	0001	1110	
US	1	15	0001	1111	1	F	0001	1111	
SP	2	0	0010	0000	4	0	0100	0000	
ı[1]	2	1	0010	0001	4	F	0100	1111	Logical OR
"	2	2	0010	0010	7	F	0111	1111	
#	2	3	0010	0011	7	B	0111	1011	
$	2	4	0010	0100	5	B	0101	1011	
%	2	5	0010	0101	6	C	0110	1100	
&	2	6	0010	0110	5	0	0101	0000	
'	2	7	0010	0111	7	D	0111	1101	
(	2	8	0010	1000	4	D	0100	1101	
)	2	9	0010	1001	5	D	0101	1101	
*	2	10	0010	1010	5	C	0101	1100	
+	2	11	0010	1011	4	E	0100	1110	
,	2	12	0010	1100	6	B	0110	1011	
-	2	13	0010	1101	6	0	0110	0000	Hyphen, Minus

Figure 56 (Page 2 of 3). ASCII to EBCDIC Correspondence

ASCII					EBCDICI				
Character	Col	Row	Bit Pattern		Col (in Hex)	Row (in Hex)	Bit Pattern		Comments
.	2	14	0010	1110	4	B	0100	1011	
/	2	15	0010	1111	6	1	0110	1001	
0	3	0	0011	0000	F	0	1111	0000	
1	3	1	0011	0001	F	1	1111	0001	
2	3	2	0011	0010	F	2	1111	0010	
3	3	3	0011	0011	F	3	1111	0011	
4	3	4	0011	0100	F	4	1111	0100	
5	3	5	0011	0101	F	5	1111	0101	
6	3	6	0011	0110	F	6	1111	0110	
7	3	7	0011	0111	F	7	1111	0111	
8	3	8	0011	1000	F	8	1111	1000	
9	3	9	0011	1001	F	9	1111	1001	
:	3	10	0011	1010	7	A	0111	1010	
;	3	11	0011	1011	5	E	0101	1110	
<	3	12	0011	1100	4	C	0100	1100	
=	3	13	0011	1101	7	E	0111	1110	
>	3	14	0011	1110	6	E	0110	1110	
?	3	15	0011	1111	6	F	0110	1111	
@	4	0	0100	0000	7	C	0111	1100	
A	4	1	0100	0001	C	1	1100	0001	
B	4	2	0100	0010	C	2	1100	0010	
C	4	3	0100	0011	C	3	1100	0011	
D	4	4	0100	0100	C	4	1100	0100	
E	4	5	0100	0101	C	5	1100	0101	
F	4	6	0100	0110	C	6	1100	0110	
G	4	7	0100	0111	C	7	1100	0111	
H	4	8	0100	1000	C	8	1100	1000	
I	4	9	0100	1001	C	9	1100	1001	
J	4	10	0100	1010	D	1	1101	0001	
K	4	11	0100	1011	D	2	1101	0010	
L	4	12	0100	1100	D	3	1101	0011	
M	4	13	0100	1101	D	4	1101	0100	
N	4	14	0100	1110	D	5	1101	0101	
O	4	15	0100	1111	D	6	1101	0110	
P	5	0	0101	0000	D	7	1101	0111	
Q	5	1	0101	0001	D	8	1101	1000	
R	5	2	0101	0010	D	9	1101	1001	
S	5	3	0101	0011	E	2	1110	0010	
T	5	4	0101	0100	E	3	1110	0011	
U	5	5	0101	0101	E	4	1110	0100	
V	5	6	0101	0110	E	5	1110	0101	
W	5	7	0101	0111	E	6	1110	0110	
X	5	8	0101	1000	E	7	1110	0111	
Y	5	9	0101	1001	E	8	1110	1000	
Z	5	10	0101	1010	E	9	1110	1001	
[	5	11	0101	1011	4	A	0100	1010	

Figure 56 (Page 3 of 3). ASCII to EBCDIC Correspondence

ASCII					EBCDICI				
Character	Col	Row	Bit Pattern		Col (in Hex)	Row (in Hex)	Bit Pattern		Comments
\	5	12	0101	1100	E	0	1110	0000	Reverse Slant
]	5	13	0101	1101	5	A	0101	1010	
¬[2]	5	14	0101	1110	5	F	0101	1111	Logical NOT
_	5	15	0101	1111	6	D	0110	1101	
`	6	0	0110	0000	7	9	0111	1001	Grave Accent
a	6	1	0110	0001	8	1	1000	0001	
b	6	2	0110	0010	8	2	1000	0010	
c	6	3	0110	0011	8	3	1000	0011	
d	6	4	0110	0100	8	4	1000	0100	
e	6	5	0110	0101	8	5	1000	0101	
f	6	6	0110	0110	8	6	1000	0110	
g	6	7	0110	0111	8	7	1000	0111	
h	6	8	0110	1000	8	8	1000	1000	
i	6	9	0110	1001	8	9	1000	1001	
j	6	10	0110	1010	9	1	1001	0001	
k	6	11	0110	1011	9	2	1001	0010	
l	6	12	0110	1100	9	3	1001	0011	
m	6	13	0110	1101	9	4	1001	0100	
n	6	14	0110	1110	9	5	1001	0101	
o	6	15	0110	1111	9	6	1001	0110	
p	7	0	0111	0000	9	7	1001	0111	
q	7	1	0111	0001	9	8	1001	1000	
r	7	2	0111	0010	9	9	1001	1001	
s	7	3	0111	0011	A	2	1010	0010	
t	7	4	0111	0100	A	3	1010	0011	
u	7	5	0111	0101	A	4	1010	0100	
v	7	6	0111	0110	A	5	1010	0101	
w	7	7	0111	0111	A	6	1010	0110	
x	7	8	0111	1000	A	7	1010	0111	
y	7	9	0111	1001	A	8	1010	1000	
z	7	10	0111	1010	A	9	1010	1001	
{	7	11	0111	1011	C	0	1100	0000	
|	7	12	0111	1100	6	A	0110	1010	Vertical Line
}	7	13	0111	1101	D	0	1101	0000	
~	7	14	0111	1110	A	1	1010	0001	Tilde
DEL	7	15	0111	1111	0	7	0000	0111	

[1] The graphic ! (Exclamation Point) can be used instead of | (Logical OR). [2] The graphic ^ (Circumflex) can be used instead of ¬ (Logical NOT).

ASCII Code

Appendix C. Standard and Non-Standard Labels

This appendix assumes that you are familiar with the concepts of label processing as described under “Job Control for Label Information” in the *VSE/ESA Guide to System Functions*. It summarizes the coding requirements in a user-written routine for the processing of user-standard labels or nonstandard labels.

Do not process IBM standard file labels in a label-processing routine of your own. The checking and writing of these labels is done by IOCS. Following is a list of IBM standard file labels that IOCS checks and writes for files on disk and on tape:

On disk:
- Format-1
- Format-2
- Format-3

On tape:
- HDR1, HDR2
- EOF1, EOF2
- EOV1, EOV2

This appendix gives the formats of labels as they are stored on volumes of data such as a tape reel or a disk pack.

Processing of User Labels

To process user-standard labels for a file, IBM standard labels for the file must exist. IOCS always processes the IBM standard label(s) for a file before it can process user-standard labels for the file.

IOCS processes IBM standard labels when one of the following occurs:

- For a file with labels, your program issues an OPEN or a CLOSE.
- IOCS finds end of file or end of volume.

IOCS assists you in the processing of user labels for a file by a routine of your own. Your program must define this routine to IOCS in the LABADDR=name operand of the DTFxx macro for the file. The types of user labels that your routine can process are:

- For a file on disk − User standard labels, header and trailer.
- For a file on tape − User standard labels, header and trailer, and nonstandard labels.

IOCS gives control to your routine:

- For a file with user-standard labels, when the processing of the IBM standard label(s) is complete.
- For a file with nonstandard labels, when IOCS opens the file or finds a tape mark.

IOCS passes to your routine:

- In the low-order byte of register 0, a code to indicate the type of label to be processed. The code is one of the following:

© Copyright IBM Corp. 1990, 1999

O = Open − Header labels are to be processed.
F = File − End-of-file labels are to be processed.
V = Volume − End-of-volume labels are to be processed.

Note: This does not apply to a disk file for whose processing you use direct access (DTFDA, DTFPH or DTFIS). It does not apply to the checking of nonstandard labels for a file on tape.

- In register 1, the address of the label to be processed; this applies to:
  - User-standard labels for a file on disk or on tape.
- In register 1, the hexadecimal representation of the symbolic unit being used (the same as in bytes 6 and 7 of the CCB); this applies to:
  - Nonstandard labels for a file on tape.

IOCS writes user labels on the first track of your file's first (or only) extent if your file is to be written onto disk. It writes the labels preceding the first data record of your file if the file is to be written onto tape.

In your label processing routine, you cannot issue a macro that calls a transient routine. Examples of these macros are OPEN, CLOSE, and CHKPT.

Coding Requirements - User-Standard Labels

Writing Your Labels on Output

- Up to eight header labels (UHL1, UHL2, and so on) and up to eight trailer labels (UTL1, UTL2, and so on) can be written for a file.
- At least a UHL1 and a UTL1 label must be written if the access method is SAM or ISAM.
- At least a UHL1 label must be written if the access method is DAM. If also trailer labels are to be written, your DTFDA for the file must include TRLBL=YES.
- The label is to be built in an 80-byte area of your program. In this area, leave the first four bytes free if your file is to be written onto disk. If your file is to be written onto tape, these four bytes must contain the label identification (UHLn or UTLn, whichever applies).
- For IOCS to write the label, provide for:
  1. Loading the address of the area into register 0.
  2. Passing control to IOCS by coding a LBRET 2 macro.

  IOCS writes the label onto the output volume. If this is a disk, IOCS inserts the correct label identification into the first four bytes. IOCS then returns control to your routine.
- When your routine has built the last label, return control to IOCS by coding a LBRET 1 macro. For a file on tape, this causes IOCS to write a tape mark.

Checking Your Labels on Input

- IOCS makes your labels available for checking one after the other, one at a time.
- If the labels are to be checked against information retrieved from another input file, that file must be opened first.
- If a label is to be updated (applies only to a file on disk), your routine must:
  1. Move the label to an area within your program.
  2. Modify the label as desired.
  3. Load the address of the modified label into register 0.
  4. Return control to IOCS by coding a LBRET 3 macro.
- After having checked a label, and if the label is not to be modified, your routine must return control to IOCS:
  - By coding a LBRET 2 macro if more labels are to be checked.
  - By coding a LBRET 1 macro if no further labels are to be checked.

Coding Requirements - Non-Standard Labels on Tape

In your label processing routine, you must use physical IOCS macros to transfer labels from virtual storage to tape and vice versa. For each label record to be transferred, your routine must (1) set up a CCB and a channel program and (2) issue an EXCP macro.

Writing Your Labels on Output

After all labels have been written, return control to IOCS by coding a LBRET 2 macro.

Checking Your Labels on Input

If the DTFMT macro for your file does not include the LABADDR=name operand, IOCS skips to the first record after the first tapemark.

When your file is opened, IOCS passes to your routine, in the low-order byte of register 0, the character O. This indicates to your routine that nonstandard header labels, if any, are to be processed.

IOCS is unable to distinguish between labels and data records. Therefore, you must read all labels and indicate the end of your checking of header labels by coding a LBRET 2 macro. For IOCS, the next tape mark is an indication of end of file or end of volume.

When IOCS reads the tapemark, your label processing routine gets control again. The routine now must:

1. Determine whether there is an end-of-file or an end-of-volume condition.
2. After all nonstandard trailer labels have been processed, return control to IOCS with a two-character condition indicator in the two low-order bytes of register 0. This indicator is:

   EF = End of file
   EV = End of volume

Formats of Volume and File Labels

The formats of labels are of interest only if an application program needs to access label information stored on a data volume.

All field displacements and lengths given in label-format descriptions are in hexadecimal notation. These displacements are relative to the beginning of each label.

Volume Label on Disk (VOL1)

Figure 57 shows the format of the DASD volume label 1 (VOL1). The volume label has a 4-byte key area and an 80-byte data area. Both the key area and the first four bytes of the data area always contain the characters "VOL1" for the first volume label. Only the data area is shown in Figure 57.

Additional volume labels, if any, are ignored by VSE.

Figure 57. Disk Volume Label (VOL1)

Displ.	Length	Content
00	04	Identifier: VOL1. Checked by IOCS. VSE supports only VOL1
04	06	Volume serial number. From EXTENT statement.
0A	01	Security byte. Used by VSE/OLTEP.
0B	05	VTOC address. Used by IOCS.
10	05	Reserved.
15	04	CI-size for FBA, blanks for CKD.
19	04	Number of blocks per CI for FBA, blank for CKD.
1D	04	Number of labels per CI for FBA, blank for CKD.
21	04	Reserved.
25	0E	Owner code for LVTOC listing.
33	1D	Reserved.

IBM Standard File Labels on Disk

There are four types of IBM standard file labels:

- **Format-1** – The normal disk file label for the first 3 extents of a file.
- **Format-2** – Used with ISAM only.
- **Format-3** – A file continuation label for additional 13 extents.
- **Format-4** – The VTOC file label, written when the disk volume is initialized.

Under VSAM, a data space is described by a file label; the characteristics of the files that occupy that space are described in the VSAM catalog. You do not name a VSAM data space; the 44-byte File-ID field contains a name assigned by VSAM. However, if a data space contains the data (or the index) of only one VSAM file (called a unique file), the File-ID field automatically contains the name you gave to the data or the index.

If a file or VSAM data space spans several volumes, the file label is repeated in the VTOC of each volume. The file label on each volume describes the portion of the file or VSAM data space on that volume and its extents.

For the formats of IBM standard file labels, refer to:

- Figure 58 − The first IBM standard disk file label (format-1).
- Figure 59 on page 445 − The IBM standard disk file continuation label (format-3).
- Figure 60 on page 445 − The VTOC file label (format-4).

  All fields of the VTOC file label are set by Device Support Facilities during volume initialization, except as indicated in Figure 60 on page 445.

Figure 58 (Page 1 of 2). IBM Standard Disk File Label (Format-1)

Displ.	Length	Content
00	2C	File ID: 1-35 bytes if generation number (Gnnn) and version number (Vnn) are specified, else 1 to 44. From DLBL or IOCS. Under VSAM, a data space name generated by the VSAM catalog routines. From VSAM routines or the VSAM DEFINE command.
2C	01	Format ID: 1. Written by IOCS on output.
2D	06	Volume serial number of first volume of the file. Written by IOCS.
33	02	Volume sequence number within the file. From IOCS.
35	03	Creation date: ydd. By IOCS from SYSCOM.
38	03	Expiration date, from DLBL or system (default: creation date + 7).
3B	01	Number of extents of the file on this volume.
3C	01	Used by MVS.
3D	01	Bit 2 = 1: expiration date specified by retention period.
3E	0D	System code: IBMDOSVS. Written by IOCS.
4B	03	Date of last access: ydd (not updated for read only disks).
4E	02	Reserved.
50	02	Number of blocks per CI for FBA, blanks for CKD.
52	02	File type (from DLBL; checked against the type of DTF): X'0008' = VSAM X'2000' = DAM X'4000' = SAM (the default) X'4100' = A library file whose file name begins with the characters IJSYS X'8000' = ISAM
54	01	Record Format. X'C0' UNDEFINED X'80' FIXED X'40' VARIABLE X'10' BLOCKED X'08' SPANNED

Figure 58 (Page 2 of 2). IBM Standard Disk File Label (Format-1)

Displ.	Length	Content
55	01	Flags for optional areas used by ISAM files (from DTF and EXTENT): Bit 2 = 1: Master index. Bit 3 = 1: Independent overflow area. Bit 4 = 1: Cylinder overflow area.
56	02	Block length. FIXED : block length UNDEFINED / VARIABLE: max block length (if present at file OPEN)
58	02	Record length. FIXED: record length UNDEFINED: X'0000' VARIABLE: max record length SPANNED: < 32K - max record length > 32K - X'8000' (if present at file OPEN)
5A	01	ISAM / DAM key length.
5B	02	Key field location in ISAM block. From DTF.
5D	01	Flags: Bit 0 = 1: Last volume (SAM only). Bit 3 = 1: File security. From DLBL.
5E	01	Original space request was: Bit 1 = 1: In blocks. Bit 4 = 1: For continuous extent. Bit 5 = 1: For maximum continuous extent. Bit 6 = 1: Not less than the specified minimum.
5F	03	Used by MVS. IOCS writes blanks.
62	05	Used by MVS. IOCS writes zeros.
67	02	Start of next record to end-of-data distance.
69	01	Type of extent (from EXTENT): X'01' = Prime data area or data space extent (the default). X'02' = Independent overflow area. X'04' = Master/cylinder index area extent. X'40' = Extent for user-standard labels. X'80' = Split cylinder extent (SAM).
6A	01	Sequence number of extent in the file. From EXTENT or IOCS.
6B	04	Extent lower limit (cchh) – from EXTENT.
6F	04	Extent upper limit (cchh) – from EXTENT.
73	0A	The same as the fields from 69 through 6F to describe a second extent, if one exists for the file.
7D	0A	The same as the fields from 69 through 6F to describe a third extent, if one exists for the file.
87	05	Address of next label for the file on this volume. Written and used by IOCS.

Figure 59. IBM Standard Disk File Continuation Label (Format-3)

Displ.	Length	Content
00	04	Key code for continuation label (03030303). Written by IOCS.
04	01	Type of extent, from EXTENT: X'01' = Data extent (default). X'80' = Split cylinder extent.
05	01	Extent sequence number.
06	04	Extent lower limit (cchh). From EXTENT.
0A	04	Extent upper limit (cchh). From EXTENT.
0E	1E	The same as the fields 04, 05, 06, and 0A, repeated three times, to describe three more extents.
2C	01	Continuation label code: EBCDIC 3, from IOCS.
2D	5A	The same as the fields 04, 05, 06, and 0A, repeated nine times, to describe nine more extents.
87	05	Address of next continuation label, if any (cchhr or 0bbbb) or zeros. From SAM IOCS only.

Figure 60 (Page 1 of 2). Disk VTOC Label (Format-4)

Displ.	Length	Content
00	2C	Key code for VTOC label: 44 times X'04'.
2C	01	VTOC label identifier: EBCDIC 4.
2D	05	Used by MVS.
32	02	Number of available file label spaces in VTOC when the volume was initialized (tracks times cylinder minus 2).
34	04	Address of next alternate track (cchh). For FBA: zeros.
38	02	Number of alternate tracks left. For FBA: zeros.
3A	01	Flags byte: Bit 0 = 1: Format-5 may not be valid (set to 1 by VSE/ESA) Bit 2 = 1: Extended free space management validity flag (set to 0 by VSE/ESA) Bit 3 = 1: Volume reserved for emulators Bit 4 = 1: Format-5 and Format-6 are accurate Bit 5 = 1: VTOC is being updated by VSAM (set to 1 by VSE/ESA) Bit 7 = 1: Indexed VTOC (set to 0 by VSE/ESA)
3B	01	Extent count. Always 1 (The VTOC is one extent).
3C	02	Reserved.
3E	0E	CKD device constants (for FBA: zeros):
3E	02	Number of cylinders
40	02	Tracks per cylinder
42	02	Length of track
44	01	Overhead byte for I (where I = a record with a key).
45	01	Overhead byte for L (where L = a last record with a key on a track).
46	01	Overhead byte for a key area.

Figure 60 (Page 2 of 2). Disk VTOC Label (Format-4)

Displ.	Length	Content
47	01	Flag byte: Bit 4 = 1: An I or L value takes two bytes. Bit 7 = 1: A tolerance is added to each record, except the last one on a track.
48	02	Tolerance. Is device-type dependent and added to the length of a record if bit 7 of the above flag byte is on.
4A	01	Number of labels on VTOC track.
4B	01	Reserved.
4C	0B	VSAM indicators (set by VSAM catalog routines):
4C	08	Time when last data space was added.
54	01	Ownership byte: Bit 0 = 1: Volume is owned by VSAM.
55	02	Number of first track of CKD catalog recovery area. For FBA: zeros.
57	09	Used by MVS.
60	04	Number of first block of FBA catalog recovery area. For CKD: zeros.
64	05	Reserved.
69	01	Extent type: 01 for VTOC extent.
6A	01	Extent sequence number: 00 = The VTOC always has one extent.
6B	04	Start address of VTOC (label).
6F	04	End address of VTOC. Used by IOCS.
73	19	Always zeros.

User-Standard File Labels on Disk

User-standard labels may be included for SAM or DAM files. VSAM and ISAM do not support them.

User-standard labels are either:

- Header labels located and processed before the data of the file, or
- Trailer labels located before and processed after the data of the file.

These labels have a 4-byte key area and an 80-byte data area. Both the key area and the first four bytes of the data area contain:

UHLn for a user header label.
UTLn for a user trailer label.

```
where: n = The label sequence number.  Can be any value
           from 1 to 8.
```

The remaining 76 bytes of the data area contain user data. Up to eight header and eight trailer labels may be written to describe a file.

Figure 61 on page 447 shows a user-standard disk file label (header and trailer).

There is always one header and one trailer label written in addition to those specified. This extra label has a 4-byte key area only and no data area. An example of five header labels and four trailer labels for a file is shown in Figure 62 on page 447.

Figure 61. User-Standard Disk-File Label (Header and Trailer)

Displ.	Length	Content
00	04	Key area: UHLn for a header label. UTLn for a trailer label. where n is the label-sequence number: A number from 1 to 8 for header labels. A number from 0 to 7 for trailer labels.
04	04	Data area: UHLn for a header label. UTLn for a trailer label. where n is the label-sequence number: a number from 1 to 8 for all user labels.
08	4C	The user's label information

Figure 62. User-Standard Disk-File Label (Five UHLs and Four UTLs are Specified)

Key Area	Data Area
UHL1	UHL1 + 76 bytes of user label data
UHL2	UHL2 + 76 bytes of user label data
UHL3	UHL3 + 76 bytes of user label data
UHL4	UHL4 + 76 bytes of user label data
UHL5	UHL5 + 76 bytes of user label data
UHL6	
UTL0	UTL1 + 76 bytes of user label data
UTL1	UTL2 + 76 bytes of user label data
UTL2	UTL3 + 76 bytes of user label data
UTL3	UTL4 + 76 bytes of user label data
UTL4	
User data	

If only header labels are specified, one UTL0 label without data is written by the system. An example in Figure 63 shows the sequence of user-standard labels with three header labels for a file.

You can include definitions or descriptions of your file in addition to those provided by the standard labels. For example, you may want to identify end-of-volume as opposed to end-of-file conditions, or you may have subcategories that you want to define for your files, or you may want to maintain an audit trail in these labels.

Figure 63. User-Standard Disk-File Label (Three UHLs are Specified)	
Key Area	**Data Area**
UHL1	UHL1 + 76 bytes of user label data
UHL2	UHL2 + 76 bytes of user label data
UHL3	UHL3 + 76 bytes of user label data
UHL4	
UTL0	
User data	

Volume Labels on Diskette

A diskette volume has one volume label of 80 bytes. It is located on track 0, sector 7 and starts with VOL1. Figure 64 shows the format of a diskette volume label.

Figure 64. Diskette Volume Label		
Displ.	**Length**	**Content**
00	04	Label ID: VOL1. The alphameric 1 in byte 3 is ignored by VSE.
04	06	Volume serial number from EXTENT.
0A	01	Accessibility indicator: S or blank. From DTF.
0B	1A	Reserved.
25	0E	Name or code of volume owner.
33	1C	Reserved.
4F	01	Label standard level: W.

IBM Standard File Labels on Diskette

The IBM standard file label on a diskette is 80 bytes long. The first four bytes of the label always contain the characters HDR1. The remaining bytes contain the start and end addresses of the file or of the extent of a file on this volume. Since only one extent of each file is on a diskette, no continuation labels are needed.

All IBM standard file labels for all files on a diskette volume are stored in the VTOC on track 0, sectors 8-26.

Only IBM standard file labels are supported on diskettes.

Figure 65 shows the format of a diskette file label.

Figure 65 (Page 1 of 2). Diskette File Label		
Displ.	**Length**	**Content**
00	04	Label ID: HDR1.
04	01	Reserved.
05	08	File-ID. From DLBL or system.
0D	09	Reserved.
16	05	Record length. From IOCS.
1B	01	Reserved.

Figure 65 (Page 2 of 2). Diskette File Label

Displ.	Length	Content
1C	05	Start address of extent: track and sector. From IOCS.
21	01	Reserved.
22	05	End address of extent: track and sector. From IOCS.
27	01	Reserved.
28	01	Bypass byte: B or blank – B = job ends on input.
29	01	Security byte: S or blank.
2A	01	Write protection byte: P or blank.

2B	01	Interchange level:
		`blank = Sector length is 128 bytes, unblocked, non-spanned, sequential.`
		`non-blank = Job ends on input.`

2C	01	Volume byte:
		`blank = File complete on this volume.`
		`C = File continued on next volume.`
		`L = File ends on this volume.`

2D	02	Volume sequence number.
2F	06	Creation date: yymmdd.
35	0D	Reserved.
42	06	Expiration date: default = seven days after output.
48	01	Verify byte: V or blank.
49	01	C 'R' expiration date specified by retention period.
4A	05	End-of-data address.
4F	01	Reserved.

Volume Labels on Tape

The volume label for tapes is 80 bytes long and begins with VOL1 for the first volume label. Additional volume labels, if any, are ignored by VSE.

Figure 66 and Figure 67 on page 450 show volume labels for EBCDIC and ASCII tapes.

Figure 66. Tape Volume Label for EBCDIC Code

Displ.	Length	Content
00	04	Label ID: VOL1.
04	06	Volume serial number.
0A	1F	Reserved.
29	0A	Volume owner name or code.
33	1D	Reserved.

Figure 67. Tape Volume Label for ASCII Code

Displ.	Length	Content
00	04	Label ID: VOL1.
04	06	Volume serial number.
0A	01	Accessibility character.
0B	1A	Reserved.
25	0E	Name or code of volume owner.
33	1C	Reserved.

4F	01	Standard byte:
		`Indicates version of label standard.`
		`If blank, file does not have ANSI standard.`

IBM Standard File Labels on Tape

IBM standard file labels are 80 bytes long. Each file has a header and a trailer label of the same format. This allows a tape to be read forward and backward.

The first four characters of a label identify its type. These characters are for:

- Header labels:

  HDR1 and HDR2 – At the beginning of a file.

- Trailer labels:

  EOF1 and EOF2 – At the end of a file.
  EOV1 and EOV2 – At the end of a volume and not the end of a file.

Data Set 2 Labels

To facilitate migration to an MVS environment, labeled output tapes created via DTFMT or DTFPH will contain standard IBM data set label 2 records (HDR2, EOF2, EOV2). The purpose of these labels is to record characteristics of the file, such as block size, record size, record format. These labels also contain information about the recording technique used to create the file, the unit on which the data was written, and the job and program (taken from the COMREG) that were used to create the file. In an MVS environment, the file can thus be used without any data control block (DCB) or data definition (DD) specification.

Data set 2 labels will also be created for standard labeled work files. On input, these labels will be skipped.

Additional labels (HDR3 through HDR8, if present) are ignored by VSE.

Figure 68 through Figure 71 on page 452 show IBM standard file labels for tapes.

Figure 68. First IBM Standard Tape File Label for EBCDIC Code

Displ.	Length	Content
00	04	Label ID: HDR1, EOF1, or EOV1.
04	11	File ID. From TLBL.
15	06	Volume serial number of first volume of the file.
1B	04	Volume sequence number within the file.
1F	04	File sequence number on the volume.
23	04	Version number of the file.
27	02	Subversion number.
29	06	Creation date: cyyddd.
2F	06	Expiration date: cyyddd.
35	01	Reserved.
36	06	Number of blocks written. Used only in trailer labels.
37	0D	System code: IBMDOSVS.
49	01	Bit 0 = 1: expiration date specified by retention period.
4A	02	Reserved.
4C	04	High order block count.

Figure 69. First IBM Standard Tape File Label for ASCII Code

Displ.	Length	Content
00	04	Label ID: HDR1, EOF1, or EOV1.
04	11	File ID. From TLBL.
15	06	Volume serial number of first volume of the file.
1B	04	Volume sequence number within the file.
1F	04	File sequence number within volume(s).
23	04	Version number of the file.
27	02	Subversion number.
29	06	Creation date: cyyddd.
2F	06	Expiration date: cyyddd.
35	01	Data set security (accessibility character).
36	06	Number of blocks written. Used only in trailer labels.
3C	0D	System code: IBMZLB, followed by two blanks.
49	07	Reserved.

Figure 70 (Page 1 of 2). Second IBM Standard Tape File Label for ASCII Code

Displ.	Length	Content
00	04	Label ID: HDR2, EOF2, or EOV2.
04	01	Record format: C'F' for fixed. C'D' for variable. C'S' for all other cases.

Figure 70 (Page 2 of 2). Second IBM Standard Tape File Label for ASCII Code

Displ.	Length	Content
05	05	Block length.
0A	05	Record length.
0F	23	Reserved.
32	02	Buffer-offset length.
34	1C	Reserved.

Figure 71 (Page 1 of 2). Second IBM Standard Tape File Label for EBCDIC Code

Displ.	Length	Content
00	04	Label ID: HDR2, EOF2, or EOV2.
04	01	Record format: **F** for fixed-length records **V** for variable-length and spanned records **U** for undefined records (default for DTFPH)
05	05	Block length, unpacked decimal (for DTFPH, a maximum block length of 32,767 is assumed).
0A	05	Record length, unpacked decimal.
0F	01	Tape density: • For 7-track tapes: **0** 200 bpi **1** 556 bpi **2** 800 bpi • For 9-track tapes: **2** 800 bpi nrzi **3** 1600 bpi pe **4** 6250 bpi gcr • For a 3480 tape this field will be blank.
10	01	Data set position: **0** No volume switch has occurred **1** A volume switch has previously occurred Open sets the field to 0 when it is the first volume, that is, whenever the volume serial number and the file serial number are the same; otherwise it sets the field to 1. For end of volume (EOV) and close (EOF), the field is set to 0 or 1 in the same way as for Open.
11	17	Job/job step identification: Bytes 1 to 8: Job name (taken from the COMREG). Byte 9: Set to '/'. Bytes 10 to 17: Program name (taken from the COMREG). For tape files that have been extended with TLBL DISP=OLD or DISP=MOD, this field in the EOV2/EOF2 labels will contain the information related to the job and job step that extended the file.

Figure 71 (Page 2 of 2). Second IBM Standard Tape File Label for EBCDIC Code

Displ.	Length	Content
22	02	Tape recording technique: • For 7-track tapes it will be set to (ƀ represents a blank): **T**ƀ Odd parity with translation. **C**ƀ Odd parity with conversion. **E**ƀ Even parity with no translation. **ET** Even parity with translation. ƀƀ Odd parity with no translation and no conversion. • For units with the Improved Data Recording feature (IDRC), the field will be set to (&blank represents a blank): **P**ƀ Data is compacted using IDRC feature. **ƀƀ** Data is noncompacted. On some data sets this field may contain NP to indicate non-compacted data. • For all other tape units this field is set to blanks.
24	01	Printer control character, always set to blanks.
25	01	Reserved for future use, set to blanks.
26	01	Block attribute. **B** for fixed-length and variable-length blocked records (RECFORM=FIXBLK or VARBLK). **S** for spanned variable-length unblocked records (RECFORM=SPNUNB). **R** for spanned variable-length blocked records (RECFORM=SPNBLK). ' ' In all other cases.
27	02	Reserved, set to blanks.
29	06	For the 3590, bytes 42-47 contain the device serial number (taken from sense byte 27-29).
2A	05	This field will be set as follows: • For the 3420, the first half byte of byte 43 contains the model number (taken from bits 4 - 7 of sense byte 17); the second half byte of byte 43 and bytes 44 - 47 contain the last four digits of the serial number of the creating tape unit (taken from sense bytes 15 and 16 as a 16-bit unsigned binary number and stored as zoned decimal equivalent). • For the 3480 and 3490, bytes 43 - 46 contain the last four digits of the serial number of the control unit (taken from sense bytes 28 and 29); byte 47 contains the device address (taken from bits 4 - 7 of sense byte 30). • For the 3424, bytes 43 - 47 contain the device serial number (taken from sense bytes 27 - 29; bits 0 - 3 of sense byte 27 are ignored).
2F	01	Checkpoint data set identifier, set to blanks.
30	20	Reserved, set to blanks.

User-Standard File Labels on Tape

User-standard labels are either header labels located and processed before the data of the file, or trailer labels located and processed after the file. These labels are identified by:

UHLn = User header label n.
UTLn = User trailer label n.

where n = A number from 1 to 8.

User-standard file labels are 80 bytes long. The first four bytes contain UHLn or UTLn, whichever applies. The remaining 76 bytes contain user data.

You can include definitions or descriptions of the file in addition to that in the IBM standard labels. You may, for example:

Have a unique numbering system for file identification.
Have subcategories that you want to define for a file.
Want to maintain an audit trail in these labels.

Non-Standard File Labels on Tape

Nonstandard labels are supported only on EBCDIC code tape labels. Nonstandard labels may have any length; they do not have a specified identification in the first four characters. These labels may contain whatever information the user desires, and in any arrangement. Programming for the processing of these labels is a user responsibility.

When files with nonstandard labels or unlabeled files are written onto a volume, its volume label (if present) is destroyed. Therefore, such files can be written only on volumes that are not expected to be used again for files with standard labels.

Appendix D. Librarian Feedback Codes

Librarian feedback codes are 1-byte values between 1 and 255. They are listed as part of the “Librarian Messages” (see *VSE/ESA Messages and Codes*) to describe unexpected conditions detected by Librarian modules as a result of externally controllable system errors.

Internal librarian feedback codes are listed in *VSE Central Functions Librarian Diagnosis Reference*, SC33-6330.

© Copyright IBM Corp. 1990, 1999

Feedback Code Dec	Hex	Description
21	15	GENERIC REQUEST WITHOUT VALID ARGUMENT
37	25	LIBINFO MISSING The connection to a library/sublibrary cannot be done because the LIBINFO is missing or the addressed library/sub-library chain does not exist.
38	26	INVALID NOTE WORD
39	27	I/O WORKAREA MISSING LIBRM GET or PUT without any BUFFER specification.
40	28	USED SERVICE FAILED A used system service returned with an unexpected return code.
43	2B	I/O ERROR
44	2C	RESOURCE ALREADY LOCKED The request was specified with LOCK=RETURN, but the resource was not available or the LOCK service failed.
45	2D	OPEN FAILURE VSAM OPEN failed, see VSAM message(s).
46	2E	CLOSE FAILURE VSAM CLOSE failed, see VSAM message(s).
48	30	MEMBER IS IN WRITE MODE (Supervisor locked)
51	33	DATA SET IS NOT A VSE LIBRARY The accessed data set is not a valid library.
52	34	NO VALID INDEX ENTRY TYPE An index block with an invalid TYPE indication has been read.
53	35	DUPLICATE TYPE ENTRY An index block with more than one TYPE entry at the beginning.
54	36	NO STARTING TYPE ENTRY IN LIBRARY BLOCK An index block must always start with a TYPE entry.
55	37	INCONSISTENCY IN MEMBER INDEX The consistency check of a library block of the member index failed.
56	38	EMPTY LIBRARY BLOCK A library block must contain at least one record.
57	39	INVALID # OF MEMBER INDEX LEVELS
59	3B	NO MBRX ENTRY AFTER TYPE ENTRY
60	3C	INVALID LB DATA INVARIANT The contents of the library block have been destroyed.

Figure 72 (Part 1 of 3). Librarian Feedback Codes

Feedback Code Dec	Hex	Description
61	3D	ZERO PRBA IN MBRX ENTRY
62	3E	LIBRARY BLOCK IDENTIFIER WRONG The library block which has been read has an LBID value which does not match the value contained in the descriptor.
63	3F	NEW DIRECTORY ENTRY EXCEEDS MAXIMUM LENGTH (255 bytes)
65	41	LIBRARY IS FULL A request for allocation of one or more LB(s) cannot be satisfied.
66	42	LIBRARY OR SUBLIBRARY CONTAINS LOCKED MEMBERS
67	43	MEMBER LOCKED
68	44	SPECIFICATION OF LOCKID IS MISSING OR INVALID
70	46	MEMBER DOES NOT EXIST A service was issued for a member which does not exist.
71	47	SUBLIBRAY DOES NOT EXIST A service was issued for a sublibrary which does not exist.
72	48	LIBRARY DOES NOT EXIST A service was issued for a library which does not exist.
100	64	GETVIS SPACE EXHAUSTED A GETVIS request failed.
105	69	LIBRARY CONTROL TABLES OVERFLOW An insertion in a library control table fails because of overflow.
106	6A	LIB/SUBLIB NOT UNIQUELY ASSIGNED
107	6B	LOGICAL UNITS EXHAUSTED The ASSGN service failed with RC=4
108	6C	VOLUME NOT MOUNTED The GETVCE service failed because the volume is not mounted.
165	A5	EXTENT ON DIFFERENT DEVICE TYPE The extents for a library cannot reside on DASDs of different types.
166	A6	INVALID OR MISSING MEMBER TYPE The library service failed due to missing or invalid member type input.
167	A7	INVALID OR MISSING MEMBER NAME
180	B4	MORE THAN 16 EXTENT STATEMENTS
181	B5	LIBRARY IS NOT EXTENDABLE
183	B7	DLBL/EXTENT STATEMENT MISSING
184	B8	INCORRECT EXTENT STATEMENT
185	B9	INCORRECT DLBL STATEMENT
186	BA	EXTENT STATEMENT MISSING

Figure 72 (Part 2 of 3). Librarian Feedback Codes

Feedback Code Dec	Hex	Description
231	E7	NO ALPHANUMERIC INPUT The library service failed because of invalid input.
232	E8	RENAME W/O NEW MEMBER INPUT The RENAME request was given without the new name (NMEMBER or NTYPE).
233	E9	NOTE STACK UNDERFLOW/OVERFLOW A NOTE stack overflow occurred during this NOTE. The maximum depth of the stack is 20. A NOTE stack underflow occurred during this POINT. The stack is empty.
234	EA	MEMBER NOT OPENED A member record operation request was issued, but the member is not opened.
235	EB	RECORD FORMAT MISMATCH The record format specified in the LDCB (librarian data control block) does not fit with the existing member or the member type specified in the LDCB for a new member LIBRM OPEN: Record format F was specified for members of type DUMP or PHASE. Record format S was specified for members of type PROC, OBJ, or source.
236	EC	INCORRECT PHASE HANDLING The request cannot handle the phase. A phase cannot be replaced or built without having correct phase-related control information.
238	EE	MISSING CHAIN-AREA The CHAIN operand is required for the issued request.
242	F2	MISSING CHAINID The CHAINID operand is required for the issued request.
246	F6	MISSING LIB or SUBLIB The LIB and SUBLIB operands must be specified together.
248	F8	INVALID MACRO SEQUENCE The applied macro sequence is invalid (e.g CLOSE before OPEN, or STATE within an OPEN/CLOSE frame).
249	F9	INVALID OPEN NESTING An OPEN for INPUT can only be issued in a nested way for Librarian-controlled NOTE and POINT requests to open a new member (stacked record inclusion from different members).
250	FA	INVALID LDCB AS INPUT A request was issued with register 1 pointing to an area which does not have the format of the LDCB.

Figure 72 (Part 3 of 3). Librarian Feedback Codes

Appendix E. VSE Macros Intended for Customer Use

The macros identified in this appendix are provided as programming interfaces for customers by VSE/ESA.

Warning: Do not use as programming interfaces any VSE/ESA macros other than those identified in this appendix.

Note: All macros mentioned in this appendix are distributed in the system library IJSYSRS.SYSLIB.

VSE/Advanced Functions

Macro Name	General Use	Product Sensitive	Described in
ALESERV	x		VSE/ESA System Macros Reference
AMODESW	x		VSE/ESA System Macros Reference
ASPL	x		VSE/ESA System Macros Reference
ASSIGN	x		VSE/ESA System Macros Reference
ATTACH	x		VSE/ESA System Macros Reference
AVRLIST	x		VSE/ESA System Macros Reference
CALL	x		VSE/ESA System Macros Reference
CANCEL	x		VSE/ESA System Macros Reference
CCB	x		VSE/ESA System Macros Reference
CDDELETE	x		VSE/ESA System Macros Reference
CDLOAD	x		VSE/ESA System Macros Reference
CHAP	x		VSE/ESA System Macros Reference
CHECK	x		VSE/ESA System Macros Reference
CHKPT	x		VSE/ESA System Macros Reference
CLOSE	x		VSE/ESA System Macros Reference
CLOSER	x		VSE/ESA System Macros Reference
CNTRL	x		VSE/ESA System Macros Reference
COMRG	x		VSE/ESA System Macros Reference
CPCLOSE	x		VSE/ESA System Macros Reference
CSRCMPSC	x		VSE/ESA System Macros Reference
DCTENTRY	x		VSE/ESA System Macros Reference
DEQ	x		VSE/ESA System Macros Reference
DETACH	x		VSE/ESA System Macros Reference
DFR	x		VSE/ESA System Macros Reference
DISEN	x		VSE/ESA System Macros Reference
DLINT	x		VSE/ESA System Macros Reference
DSPLY	x		VSE/ESA System Macros Reference
DSPSERV	x		VSE/ESA System Macros Reference
DTFCD	x		VSE/ESA System Macros Reference
DTFCN	x		VSE/ESA System Macros Reference
DTFDA	x		VSE/ESA System Macros Reference
DTFDI	x		VSE/ESA System Macros Reference
DTFDR	x		VSE/ESA System Macros Reference
DTFDU	x		VSE/ESA System Macros Reference
DTFIS	x		VSE/ESA System Macros Reference
DTFMR	x		VSE/ESA System Macros Reference
DTFMT	x		VSE/ESA System Macros Reference
DTFOR	x		VSE/ESA System Macros Reference
DTFPH	x		VSE/ESA System Macros Reference
DTFPR	x		VSE/ESA System Macros Reference

© Copyright IBM Corp. 1990, 1999

Macro Name	General Use	Product Sensitive	Described in
DTFSD	x		VSE/ESA System Macros Reference
DTL	x		VSE/ESA System Macros Reference
DUMP	x		VSE/ESA System Macros Reference
ENDFL	x		VSE/ESA System Macros Reference
ENQ	x		VSE/ESA System Macros Reference
EOJ	x		VSE/ESA System Macros Reference
ERET	x		VSE/ESA System Macros Reference
ESETL	x		VSE/ESA System Macros Reference
EXCP	x		VSE/ESA System Macros Reference
EXIT	x		VSE/ESA System Macros Reference
EXTRACT	x		VSE/ESA System Macros Reference
FCEPGOUT	x		VSE/ESA System Macros Reference
FEOV	x		VSE/ESA System Macros Reference
FEOVD	x		VSE/ESA System Macros Reference
FETCH	x		VSE/ESA System Macros Reference
FREE	x		VSE/ESA System Macros Reference
FREEVIS	x		VSE/ESA System Macros Reference
GENDTL	x		VSE/ESA System Macros Reference
GENIORB	x		VSE/ESA System Macros Reference
GENL	x		VSE/ESA System Macros Reference
GET	x		VSE/ESA System Macros Reference
GETIME	x		VSE/ESA System Macros Reference
GETSYMB	x		VSE/ESA System Macros Reference
GETVCE	x		VSE/ESA System Macros Reference
GETVIS	x		VSE/ESA System Macros Reference
IJBPUB	x		VSE/ESA System Macros Reference
IORB	x		VSE/ESA System Macros Reference
JDUMP	x		VSE/ESA System Macros Reference
JOBCOM	x		VSE/ESA System Macros Reference
LBRET	x		VSE/ESA System Macros Reference
LBSERV	x		VSE/ESA System Macros Reference
LFCB	x		VSE/ESA System Macros Reference
LIBRDCB	x		VSE/ESA System Macros Reference
LIBRM xxx	x		VSE/ESA System Macros Reference
LITE	x		VSE/ESA System Macros Reference
LOAD	x		VSE/ESA System Macros Reference
LOCK	x		VSE/ESA System Macros Reference
MAPBDY	x		VSE/ESA System Macros Reference
MAPBDYVR	x		VSE/ESA System Macros Reference
MAPDNTRY	x		VSE/ESA System Macros Reference
MAPEXTR	x		VSE/ESA System Macros Reference
MAPSAVAR	x		VSE/ESA System Macros Reference
MAPSSID	x		VSE/ESA System Macros Reference
MAPSYSP	x		VSE/ESA System Macros Reference
MAPXPCCB	x		VSE/ESA System Macros Reference
MODDTL	x		VSE/ESA System Macros Reference
MVCOM	x		VSE/ESA System Macros Reference
NOTE	x		VSE/ESA System Macros Reference
OPEN	x		VSE/ESA System Macros Reference
OPENR	x		VSE/ESA System Macros Reference
PAGEIN	x		VSE/ESA System Macros Reference
PDUMP	x		VSE/ESA System Macros Reference
PFIX	x		VSE/ESA System Macros Reference
PFREE	x		VSE/ESA System Macros Reference
POINTR	x		VSE/ESA System Macros Reference
POINTS	x		VSE/ESA System Macros Reference
POINTW	x		VSE/ESA System Macros Reference

Macro Name	General Use	Product Sensitive	Described in
POST	x		VSE/ESA System Macros Reference
PRTOV	x		VSE/ESA System Macros Reference
PUT	x		VSE/ESA System Macros Reference
PUTR	x		VSE/ESA System Macros Reference
QSETPRT	x		VSE/ESA System Macros Reference
RCB	x		VSE/ESA System Macros Reference
RDLNE	x		VSE/ESA System Macros Reference
READ	x		VSE/ESA System Macros Reference
REALAD	x		VSE/ESA System Macros Reference
RELEASE	x		VSE/ESA System Macros Reference
RELPAG	x		VSE/ESA System Macros Reference
RELSE	x		VSE/ESA System Macros Reference
RESCN	x		VSE/ESA System Macros Reference
RETURN	x		VSE/ESA System Macros Reference
RUNMODE	x		VSE/ESA System Macros Reference
SAVE	x		VSE/ESA System Macros Reference
SDUMP	x		VSE/ESA System Macros Reference
SDUMPX	x		VSE/ESA System Macros Reference
SECTVAL	x		VSE/ESA System Macros Reference
SEOV	x		VSE/ESA System Macros Reference
SETDEV	x		VSE/ESA System Macros Reference
SETFL	x		VSE/ESA System Macros Reference
SETIME	x		VSE/ESA System Macros Reference
SETL	x		VSE/ESA System Macros Reference
SETPFA	x		VSE/ESA System Macros Reference
SETPRT	x		VSE/ESA System Macros Reference
SPLEVEL	x		VSE/ESA System Macros Reference
STXIT	x		VSE/ESA System Macros Reference
SUBSID	x		VSE/ESA System Macros Reference
SYSSTATE	x		VSE/ESA System Macros Reference
TECB	x		VSE/ESA System Macros Reference
TPIN	x		VSE/ESA System Macros Reference
TPOUT	x		VSE/ESA System Macros Reference
TRUNC	x		VSE/ESA System Macros Reference
TTIMER	x		VSE/ESA System Macros Reference
UNLOCK	x		VSE/ESA System Macros Reference
VIRTAD	x		VSE/ESA System Macros Reference
WAIT	x		VSE/ESA System Macros Reference
WAITF	x		VSE/ESA System Macros Reference
WAITM	x		VSE/ESA System Macros Reference
WRITE	x		VSE/ESA System Macros Reference
WTO	x		VSE/ESA System Macros Reference
WTOR	x		VSE/ESA System Macros Reference
XECBTAB	x		VSE/ESA System Macros Reference
XPCC	x		VSE/ESA System Macros Reference
XPCCB	x		VSE/ESA System Macros Reference
XPOST	x		VSE/ESA System Macros Reference
XWAIT	x		VSE/ESA System Macros Reference

VSE/SP Unique Code

Macro Name	General Use	Product Sensitive	Described in
INWMAPI	x		*VSE/ESA Programming and Workstation Guide*

VSE/POWER

Execution Macros

Macro Name	General Use	Product Sensitive	Described in
CTLSPOOL		x	VSE/POWER Application Programming
GETSPOOL		x	VSE/POWER Application Programming
PUTACCT		x	VSE/POWER Application Programming
PUTSPOOL		x	VSE/POWER Application Programming
PWRSPL		x	VSE/POWER Application Programming
SPL		x	VSE/POWER Application Programming
SEGMENT		x	VSE/POWER Application Programming
IPWSEGM		x	VSE/POWER Application Programming
XECBTAB	x		VSE/POWER Application Programming
XPCC	x		VSE/POWER Application Programming
XPCCB	x		VSE/POWER Application Programming

Mapping Macros

Macro Name	General Use	Product Sensitive	Described in
IPW$MXD		x	VSE/POWER Application Programming
IPW$DXE		x	VSE/POWER Application Programming
IPW$DTX		x	VSE/POWER Networking
IPW$IDM		x	VSE/ICCF Administration and Operation
MAPXPCCB	x		VSE/POWER Application Programming
PACCNT	x		VSE/POWER Application Programming
PWRSPL	x		VSE/POWER Application Programming
SPL	x		VSE/POWER Application Programming

VSE/Interactive Computing and Control Facility (VSE/ICCF)

Macro Name	General Use	Product Sensitive	Described in
A$HELP	x		VSE/ICCF User's Guide
A$MAIL	x		VSE/ICCF User's Guide
ASSEMBLE	x		VSE/ICCF User's Guide
COBOL	x		VSE/ICCF User's Guide
COPY	x		VSE/ICCF User's Guide

Macro Name	General Use	Product Sensitive	Described in
COPYFILE	x		VSE/ICCF User's Guide
COPYMEM	x		VSE/ICCF User's Guide
CPYLIB	x		VSE/ICCF User's Guide
DTSGENER		x	VSE/ICCF Administration and Operation
DTSGOFS	x		VSE/ICCF Administration and Operation
DTSLG00	x		VSE/ICCF User's Guide
DTSLG4R	x		VSE/ICCF User's Guide
DTSLG40	x		VSE/ICCF User's Guide
DTSLG7A	x		VSE/ICCF User's Guide
DTSLG70	x		VSE/ICCF User's Guide
DTSOPTNS		x	VSE/ICCF Administration and Operation
DTSSCRN	x		VSE/ICCF User's Guide
DTSWRTRD	x		VSE/ICCF User's Guide
ED	x		VSE/ICCF User's Guide
EDPRT	x		VSE/ICCF User's Guide
EDPUN	x		VSE/ICCF User's Guide
FORTPROG	x		VSE/ICCF User's Guide
FORTRAN	x		VSE/ICCF User's Guide
FSEDPF	x		VSE/ICCF User's Guide
GETL	x		VSE/ICCF User's Guide
GETP	x		VSE/ICCF User's Guide
GETR	x		VSE/ICCF User's Guide
HC	x		VSE/ICCF User's Guide
HELP	x		VSE/ICCF User's Guide
HELP$LIS	x		VSE/ICCF User's Guide
HELP$LST	x		VSE/ICCF User's Guide
LIBRC	x		VSE/ICCF User's Guide
LIBRL	x		VSE/ICCF User's Guide
LIBRP	x		VSE/ICCF User's Guide
LOAD	x		VSE/ICCF User's Guide
MOVE	x		VSE/ICCF User's Guide
MVLIB	x		VSE/ICCF User's Guide
PLI	x		VSE/ICCF User's Guide
PRINT	x		VSE/ICCF User's Guide
RELIST	x		VSE/ICCF User's Guide
RPGIAUTO	x		VSE/ICCF User's Guide
RPGII	x		VSE/ICCF User's Guide
RPGIXLTR	x		VSE/ICCF User's Guide
RSEF	x		VSE/ICCF User's Guide
SAMPASMB	x		VSE/ICCF User's Guide
SAMPBASC	x		VSE/ICCF User's Guide
SAMPCOB	x		VSE/ICCF User's Guide
SAMPFORT	x		VSE/ICCF User's Guide
SAMPPLI	x		VSE/ICCF User's Guide
SCHD1ASM	x		VSE/ICCF User's Guide
SCHD2ASM	x		VSE/ICCF User's Guide
SCHD1COB	x		VSE/ICCF User's Guide
SCHD2COB	x		VSE/ICCF User's Guide
SCHD1PLI	x		VSE/ICCF User's Guide
SCHD2PLI	x		VSE/ICCF User's Guide
SCRATCH	x		VSE/ICCF User's Guide
SDSERV	x		VSE/ICCF User's Guide
SORT	x		VSE/ICCF User's Guide
STORE	x		VSE/ICCF User's Guide
SUBMIT	x		VSE/ICCF User's Guide
VSBASIC	x		VSE/ICCF User's Guide
VSBRESEQ	x		VSE/ICCF User's Guide

Macro Name	General Use	Product Sensitive	Described in
WORKFILE	x		VSE/ICCF User's Guide
WORKFIL2	x		VSE/ICCF User's Guide
WORKFIL3	x		VSE/ICCF User's Guide

VSE/Virtual Storage Access Method (VSE/VSAM)

Macro Name	General Use	Product Sensitive	Described in
ACB	x		*VSE/VSAM User's Guide and Application Programming*
BLDVRP	x		*VSE/VSAM User's Guide and Application Programming*
DLVRP	x		*VSE/VSAM User's Guide and Application Programming*
ENDREQ	x		*VSE/VSAM User's Guide and Application Programming*
ERASE	x		*VSE/VSAM User's Guide and Application Programming*
EXLST	x		*VSE/VSAM User's Guide and Application Programming*
GENCB	x		*VSE/VSAM User's Guide and Application Programming*
MODCB	x		*VSE/VSAM User's Guide and Application Programming*
POINT	x		*VSE/VSAM User's Guide and Application Programming*
RPL	x		*VSE/VSAM User's Guide and Application Programming*
SHOWCAT	x		*VSE/VSAM User's Guide and Application Programming*
SHOWCB	x		*VSE/VSAM User's Guide and Application Programming*
TCLOSE	x		*VSE/VSAM User's Guide and Application Programming*
TESTCB	x		*VSE/VSAM User's Guide and Application Programming*
WRTBFR	x		*VSE/VSAM User's Guide and Application Programming*

Appendix F. VSE/ESA Macros And Their Mode Dependencies

Figure 73 on page 466 lists the VSE/ESA macros and the modes allowed at execution time. The list mainly applies to the 31-bit addressing support, except for the AR MODE column which applies to the data space support.

- AMODE indicates the *addressing mode* that is expected to be in effect when the program is entered. AMODE can have one of the following values:

  AMODE 24 Specifies 24-bit addressing mode
  AMODE 31 Specifies 31-bit addressing mode

  An **X** in column AMODE 24 or AMODE 31 indicates that the macro can be invoked by a program executing in that mode. An **X** in both AMODE 24 and AMODE 31 columns implies AMODE ANY which indicates that the macro can be used and executed in both 24-bit and 31-bit addressing mode.

- RMODE indicates the *residency mode*, that is, the virtual storage location (either below 16MB or anywhere in virtual storage) where the program should reside. RMODE can have one of the following values:

  RMODE 24 Indicates that the program must reside in virtual storage below 16MB.
  RMODE ANY Indicates that the program can reside anywhere in virtual storage, either above or below 16MB.

  An **X** in column RMODE 24 or RMODE ANY indicates that the macro can be invoked by a program with the corresponding mode.

  The parameter list of a requested macro must have the same RMODE as the macro itself. The most important exceptions are pointed out in the 'Comments' column; for more details, see the corresponding macro description.

- AR MODE: Most macros listed in Figure 73 on page 466 can only be called in *primary* ASC (address space control) mode, except those that have an indication for *Access Register (AR)* ASC mode in column **AR MODE** (for macros supporting data spaces). An **X** in that column indicates that both primary and AR mode are possible.

Most macro services based on branch and link interfaces do **not** check for execution mode violations, that is, the program requesting the macro service is responsible for the correct execution mode (AMODE and RMODE). An execution mode violation may lead to unpredictable results.

Important

The following list reflects the VSE/ESA macro support at the time this manual was printed. For the latest status of the actual macro support provided, consult the *Program Directory* which is part of the VSE/ESA shipment package.

© Copyright IBM Corp. 1990, 1999

Macro	AMODE 24	AMODE 31	RMODE 24	RMODE ANY	AR MODE	Comments
ALESERV	X	X	X	X	X	
AMODESW	X	X	X	X		
ASPL			X			
ASSIGN	X		X			
ATTACH	X	X	X	X		Subtask save area RMODE=24
AVRLIST			X			
CALL	X	X	X	X		
CANCEL	X	X	X	X		
CCB			X			
CDDELETE	X	X	X	X		
CDLOAD	X	X	X	X		
CDMOD	X		X			
CHAP	X	X	X	X		
CHECK	X		X			
CHKPT	X		X			
CLOSE	X	X	X			DTF has to be allocated below 16MB
CNTRL	X		X			
COMRG	X	X	X	X		
CPCLOSE	X		X			
DCTENTRY			X			
DEQ	X	X	X	X		
DETACH	X	X	X	X		Subtask save area RMODE=24
DFR			X			
DIMOD	X		X			
DISEN	X		X			
DLINT			X			
DRMOD	X		X			
DSPLY	X		X			
DSPSERV		X	X	X	X	AR mode: SYSSTATE necessary
DTFxx			X			
DTL			X			
DUMODFx	X		X			
DUMP	X	X	X			
ENDFL	X		X			
ENQ	X	X	X	X		
EOJ	X	X	X	X		
EOJ RC=	X	X	X			
ERET	X		X			
ESETL	X		X			
EXCP	X	X	X	X		Control blocks RMODE=24
EXIT	X	X	X	X		
EXTRACT	X	X	X	X		
FCEPGOUT	X		X			SPLEVEL SET=1
	X	X	X	X		SPLEVEL SET>1
FEOV	X		X			
FEOVD	X		X			
FETCH	X	X	X			
FREE	X		X			

Figure 73 (Part 1 of 3). VSE/ESA Macros and Their Mode Dependencies (Execution Time)

Macro	AMODE 24	AMODE 31	RMODE 24	RMODE ANY	AR MODE	Comments
FREEVIS	X	X	X	X		
GENDTL	X	X	X			
GENIORB	X		X			
GENL	X	X	X			
GET	X		X			
GETIME	X	X	X	X		
GETSYMB	X		X			
GETVCE	X	X	X			
GETVIS	X	X	X	X		New parameter LOC=
IORB			X			
ISMOD	X		X			
JDUMP	X	X	X			
JOBCOM	X		X			
LBRET	X		X			
LFCB	X		X			
LIBRDCB	X		X			
LIBRM	X		X			
LITE	X		X			
LOAD	X	X	X			
LOCK	X	X	X			
MAPBDY			X	X		
MAPBDYVR			X	X		
MAPEXTR			X	X		
MAPSYSP			X	X		
MODDTL	X	X	X			
MRMOD	X		X			
MVCOM	X		X			
NOTE	X		X			
OPEN	X	X	X			DTF has to be allocated below 16MB
ORMOD	X		X			
PAGEIN	X		X			SPLEVEL SET=1
	X	X	X	X		SPLEVEL SET>1
PDUMP	X	X	X			
PFIX	X		X			SPLEVEL SET=1
	X	X	X	X		SPLEVEL SET>1
PFREE	X		X			SPLEVEL SET=1
	X	X	X	X		SPLEVEL SET>1
POINTR	X		X			
POINTS	X		X			
POINTW	X		X			
POST	X	X	X	X		
PRMOD	X		X			
PRTOV	X		X			

Figure 73 (Part 2 of 3). VSE/ESA Macros and Their Mode Dependencies (Execution Time)

Macro	AMODE 24	AMODE 31	RMODE 24	RMODE ANY	AR MODE	Comments
PUT	X		X			
PUTR	X		X			
RCB			X	X		
RDLNE	X		X			
READ	X		X			
REALAD	X	X	X	X		
RELEASE	X		X			
RELPAG	X		X			SPLEVEL SET=1
	X	X	X	X		SPLEVEL SET>1
RELSE	X		X			
RESCN	X		X			
RETURN	X	X	X	X		Returns with mode of issuer
RUNMODE	X		X			
SAVE	X	X	X	X		
SDUMP(X)	X	X	X	X	X	AR mode: SYSSTATE necessary
SECTVAL	X	X	X	X		
SEOV	X		X			
SETDEV	X		X			
SETFL	X		X			
SETIME	X	X	X	X		
SETL	X		X			
SETPFA	X		X			
STXIT	X	X	X	X		New parameter AMODE=
SUBSID	X	X	X	X		
SYSSTATE	X	X	X	X		
TECB			X	X		
TPIN	X		X			
TPOUT	X		X			
TRUNC	X		X			
TTIMER	X	X	X	X		
UNLOCK	X	X	X			
VIRTAD	X	X	X	X		
WAIT	X	X	X	X		BTAM: AMODE=24 only
WAITF	X		X			
WAITM	X	X	X	X		New ECB list in AMODE ANY
WRITE	X		X			
WTO	X	X	X	X		
WTOR	X	X	X			
XECBTAB	X		X			
XPCC	X	X	X	X		
XPCCB			X	X		
XPOST	X		X			
XWAIT	X		X			

Figure 73 (Part 3 of 3). VSE/ESA Macros and Their Mode Dependencies (Execution Time)

VSE/ESA Downward-Compatible Macros

The following macros are downward compatible to VSE/ESA 1.1, VSE/ESA 1.2 and VSE/ESA 1.3. Applications using these macros can be assembled with the VSE/ESA 2.1 macro library and can run on previous VSE/ESA Version 1 levels (1.1, 1.2, 1.3).

AVRLIST
CALL
CANCEL
CCB
CDDELETE[1]
CDLOAD
CDMOD
CHECK
CHKPT
CLOSE
CLOSER
CNTRL
COMRG
DCTENTRY
DIMOD
DTFCN
DTFDI
DTFDR
DTFDU
DTFPH
DTFSD
DUMODFI
DUMODFO
DUMP
EOJ
ERET
EXIT
EXTRACT
FEOV
FEOVD
FREEVIS
GET
GETIME
GETVCE
GETVIS[2]
IJBPUB
JDUMP
LBRET
IBRDCB
LIBRM
LOAD
MAPBDY
MAPBDYVR
MAPSSID
NOTE
OPEN
OPENR
PDUMP
POINTR
POINTS
POINTW
PRMOD
PUT
PUTR
READ
RETURN
SAVE
STXIT[3]
SUBSID
TRUNC
WAIT
WRITE
WTO[4]
WTOR[4]
XPCC[3]

[1] Interpreted like a CDLOAD in VSE/ESA 1.1, 1.2.

[2] LOC=ANY operand is accepted, but ignored in VSE/ESA 1.1, 1.2.

[3] If no new functions (introduced with VSE/ESA 1.3) are used.

[4] If no new functions (introduced with VSE/ESA 2.1) are used. Refer to the description of the “WTO (Write to Operator) Macro” on page 403, the “WTOR (Write to Operator with Reply) Macro” on page 410, and the “SPLEVEL (Set and Test Macro Level) Macro” on page 382.

Glossary

This glossary includes terms and definitions related primarily to IBM VSE/ESA.

If you do not find the term you are looking for, refer to the index of this book or to the *IBM Dictionary of Computing, SC20-1699*

The glossary includes definitions with:

- Symbol * where there is a one-to-one copy from the IBM Dictionary of Computing.
- Symbol (A) from the *American National Dictionary for Information Processing Systems*, copyright 1982 by the Computer and Business Equipment Manufacturers Association (CBEMA). Copies may be purchased from the American National Standards Institute, 1430 Broadway, New York, New York 10018. Definitions are identified by the symbol (A) after the definition.
- Symbols (I) or (T) from the *ISO Vocabulary - Information Processing* and the *ISO Vocabulary - Office Machines,* developed by the International Organization for Standardization, Technical Committee 97, Subcommittee 1. Definitions of published segments of the vocabularies are identified by the symbol (I) after the definition; definitions from draft international standards, draft proposals, and working papers in development by the ISO/TC97/SC1 vocabulary subcommittee are identified by the symbol (T) after the definition, indicating final agreement has not yet been reached among participating members.

The following cross-references are used:

- Contrast with. This refers to a term that has an opposed or substantively different meaning.
- Synonym for. This indicates that the term has the same meaning as a preferred term, which is defined in its proper place in the dictionary.
- Synonymous with. This is a backward reference from a defined term to all other terms that have the same meaning.
- See. This refers the reader to multiple-word terms that have the same last word.
- See also. This refers the reader to related terms that have a related, but not synonymous, meaning.

When an entry is an abbreviation, the explanation consists of the spelled-out meaning of the abbreviation, for example:

```
CI.  Control interval.
```

The spelled-out form is provided as a separate entry in the glossary. In that entry, the abbreviation is shown in parentheses after the spelled-out form. The definition that appears with the spelled-out entry provides the full meaning of both the abbreviation and the spelled-out form:

```
Control interval (CI).  A fixed-length area...
```

access control. A function of VSE that ensures that the system and the data and programs stored in it can be accessed only by authorized users in authorized ways.

access list. A table in which each entry specifies an address space or data space that a program can reference.

access method. A program, that is, a set of commands (macros), to define files or addresses and to move data to and from them; for example VSE/VSAM or VSE/VTAM.

access register (AR). A hardware register that a program can use to identify an address space or a data space. Each processor has 16 ARs, numbered 0 through 15, which are paired one-to-one with the 16 general-purpose registers (GPRs).

ACF/VTAM. See VTAM.

addressing mode (AMODE). A program attribute that refers to the address length that a program is prepared to handle on entry. Addresses may be either 24 bits or 31 bits in length. In 24-bit addressing mode, the processor treats all virtual addresses as 24-bit values; in 31-bit addressing mode, the processor treats all virtual addresses as 31-bit values. Programs with an addressing mode of ANY can receive control in either 24-bit or 31-bit addressing mode.

address space. A range of up to two gigabytes of contiguous virtual storage addresses that the system creates for a user. Unlike a data space, an address space contains user data **and** programs, as well as system data and programs, some of which are common to all address spaces. Instructions execute in an address space (not a data space). Contrast with data space.

address space control (ASC) mode. The mode (determined by the PSW) that tells the system where to find referenced data. It determines how the processor resolves address references for the executing programs. VSE/ESA supports two types of ASC modes:

© Copyright IBM Corp. 1990, 1999

1. In **primary** ASC mode, the data that a program can access resides in the program's own (primary) address space. In this mode, the system uses the contents of general-purpose registers to resolve an address in the address space; it does not use the contents of the access registers (ARs).
2. In **access register (AR)** ASC mode, the data that a program can access may reside in an address space other than the primary or in a data space. In this mode, the system uses both a general-purpose register (GPR) and the corresponding access register together to resolve an address in another address space or in a data space. Specifically, the AR contains a value, called an ALET, that identifies the address space or data space that contains the data, and the GPR contains a base address that points to the data within the address space or data space.

ALET (access list entry token). A token that points to an entry in an access list. When a program is in AR mode and the ALET is in an access register (with the corresponding general-purpose register being used as base register), the ALET identifies the address space or data space that the system is to reference (while the GPR indicates the offset within the space).

*** alternate tape**. A tape drive to which the operating system switches automatically for tape read or write operations if the end of the volume has been reached on the originally used tape drive.

*** alternate track**. On a direct access device, a track designated to contain data in place of a defective primary track.

*** American National Standard Code for Information Interchange (ASCII)**. The standard code, using a coded character set consisting of 7-bit coded characters (8 bits including parity check), that is used for information interchange among data processing systems, data communication systems, and associated equipment. The ASCII set consists of control characters and graphic characters. (A)

AMODE. Addressing mode.

*** appendage routine**. Code physically located in a program or subsystem, but logically an extension of a VSE supervisor routine.

*** application program**. A program written for or by a user that applies directly to the user's work, such as a program that does inventory control or payroll. See also *batch program* and *online application program*.

AR (access register) mode. If a program runs in AR mode, the system uses the access register/general-purpose register pair to resolve an address in an address space or data space. Contrast with *primary mode*. See also *address space control (ASC) mode*.

ASC mode. Address space control mode.

ASCII. American National Standard Code for Information Interchange.

*** assemble**. To translate an assembly language program into an object program. (T)

*** assembler**. A computer program that converts assembly language instructions into object code.

assembler language. A programming language whose instructions are usually in one-to-one correspondence with machine instructions and allows to write macros.

batch program. A program that is processed in series with other programs and therefore normally processes data without user interaction.

block. Usually, a block consists of several records of a file that are transmitted as a unit. But if records are very large, a block can also be part of a record only. On an FBA disk, a block is a string of 512 bytes of data. See also *control block*.

blocking. The process of combining (or cutting) records into blocks.

*** BTAM-ES (Basic Telecommunication Access Method Extended Storage)**. An IBM supplied telecommunication access method that permits read and write communication with remote devices.

*** catalog**. 1. A directory of files and libraries, with reference to their locations. A catalog may contain other information such as the types of devices in which the files are stored, passwords, blocking factors. (I) (A) 2. To store a library member such as a phase, module, or book in a sublibrary.

*** cataloged procedure**. A set of control statements placed in a library and retrievable by name.

*** catalog recovery area (CRA)**. In systems with VSAM, an entry-sequenced data set that exists on each volume owned by a recoverable catalog, including the catalog volume itself. The CRA contains copies of the catalog records and can be used to recover a damaged or invalid catalog.

CCB. Command control block.

cell pool. An area of virtual storage obtained by an application program and managed by the callable cell pool services. A cell pool is located in an address space or a data space and contains an anchor, at least one extent, and any number of cells of the same size.

*** chaining**. A logical connection of sublibraries to be searched by the system for members of the same type; for example, phase or object modules.

*** channel program**. One or more channel command words that control a sequence of data channel operations. Execution of this sequence is initiated by a single start I/O (SIO) instruction.

character printer. A device that prints a single character at a time. (T) (A) Contrast with *line printer, page printer*.

*** checkpoint**. 1. A point at which information about the status of a job and the system can be recorded so that the job step can be restarted later. 2. To record such information.

CI. Control interval.

CKD device. Count-key-data device.

*** cluster controller**. A device that can control the input/output operations of more than one device connected to it. A cluster controller may be run by a program stored and executed in the unit; for example, the IBM 3601 Finance Communication Controller. Or it may be entirely controlled by hardware; for example, the IBM 3272 Control Unit.

command control block (CCB). The name of a system control block to hold information about a specific instance of a command.

*** communication controller**. 1. A device that directs the transmission of data over the data links of a network; its operation may be controlled by a program executed in a processor to which the controller is connected or it may be controlled by a program executed within the device. (T) 2. A type of communication control unit whose operations are controlled by one or more programs stored and executed in the unit. It manages the details of line control and the routing of data through a network.

*** communication region**. An area of the supervisor that is set aside for transfer of information within and between programs.

*** compile**. To translate a source program into an executable program (an object program). See also *assembler*.

component. 1. Hardware or software that is part of a computer system. 2. A functional part of a product, identified by a component identifier. 3. In VSE/VSAM, a named, cataloged group of stored records, such as the data component or index component of a key-sequenced file or alternate index.

*** configuration**. The devices and programs that make up a system, subsystem, or network.

connect. To authorize library access on the lowest level. A modifier such as "read" or "write" is required for the specified use of a sublibrary.

control block. An area within a program or a routine defined for the purpose of storing and maintaining control information.

*** control interval (CI)**. A fixed-length area of disk storage where VSE/VSAM stores records and distributes free space. It is the unit of information that VSE/VSAM transfers to or from disk storage. For FBA, it must be an integral multiple, to be defined at cluster definition, of the block size.

control program. A program to schedule and supervise the running of programs in a system.

control unit. See *communication controller* and *cluster controller*. Synonymous with *controller*.

count-key-data (CKD) device. A disk device that stores data in the record format: count field, key field, data field. The count field contains, among others, the address of the record in the format: cylinder, head (track), record number and the length of the data field. The key field, if present, contains the record's key or search argument. CKD disk space is allocated by tracks and cylinders. Contrast with *FBA disk device*. See also *extended count-key-data device*.

CRA. Catalog recovery area.

*** cross-partition communication control**. A facility that enables VSE subsystems and user programs to communicate with each other; for example, with VSE/POWER.

DASD sharing. An option that lets independent computer systems use common data on shared disk devices.

data link. In SNA, the combination of the link connection and the link stations joining network nodes, for example, a System/370 channel and its associated protocols. A link is both logical and physical. In SNA, synonym for *link*.

*** data management**. A major function of the operating system. It involves organizing, storing, locating, and retrieving data.

data set. See *file*.

data space. A range of up to two gigabytes of contiguous virtual storage addresses that a program can directly manipulate through ESA/370 instructions. Unlike an address space, a data space can hold only

user data; it does not contain shared areas, system data or programs. Instructions do not execute in a data space, although a program can reside in a data space as non-executable code. Contrast with address space.

*** deblocking**. The process of making each logical record of a block available for processing. Contrast with *blocking*.

default value. A value assumed by the program when no value has been specified by the user.

*** device address**. 1. The identification of an input/output device by its channel and unit number. 2. In data communication, the identification of any device to which data can be sent or from which data can be received.

*** device class**. The generic name for a group of device types; for example, all display stations belong to the same device class. Contrast with *device type*.

*** Device Support Facilities**. An IBM supplied system control program for performing operations on disk volumes so that they can be accessed by IBM and user programs. Examples of these operations are initializing a disk volume and assigning an alternate track.

*** device type code**. The four- or five-digit code to be used for defining an I/O device to a computer system.

direct access. Accessing data on a storage device using their address and not their sequence. This is the typical access on disk devices as opposed to magnetic tapes. Contrast with *sequential access*.

directory. 1. A table of identifiers and references to the corresponding items of data. (I) (A) 2. In VSE, specifically, the index for the program libraries. See also *library directory* and *sublibrary directory*.

DU-AL (dispatchable unit - access list). The access list that is associated with a VSE/ESA main task or subtask. A program uses the DU-AL associated with its task and the PASN-AL associated with its partition. See also PASN-AL.

dynamic partition. A partition created and activated on an 'as needed' basis that does not use fixed static allocations. After processing, the occupied space is released. Contrast with *static partition*.

EBCDIC. Extended binary-coded decimal interchange code.

ECKD device. Extended count-key-data device.

Enterprise Systems Architecture (ESA). See *ESA/370* and *ESA/390*.

ESA mode. Such a supervisor will run on Enterprise Systems Architecture processors (ESA/370 and ESA/390) and provides support for multiple virtual address spaces, the channel subsystem, and more than 16MB of real storage.

ESA/370. IBM Enterprise Systems Architecture/370. The extension to the IBM System/370 architecture which includes the advanced addressability feature that provides access registers.

ESA/390. IBM Enterprise Systems Architecture/390. The latest extension to the IBM System/370 architecture which includes the advanced addressability feature and advanced channel architecture.

*** escape**. To return to the original level of a user interface.

extended addressability. 1. See *31-bit addressing*. 2. The ability of a program to use virtual storage that is outside the address space in which the program is running. Generally, instructions and data reside in a single address space - the primary address space. However, a program can have data in address spaces other than the primary or in data spaces. (The instructions remain in the primary address space, whilst the data can reside in another address space or in a data space.) To access data in other address spaces, a program must use access registers (ARs) and execute in access register mode (AR mode).

extended binary-coded decimal interchange code (EBCDIC). A coded character set consisting of 8-bit coded characters.

extended count-key-data (ECKD) device. A disk storage device that has a data transfer rate faster than some processors can utilize. A specialized channel program is needed to convert ordinary CKD channel programs for use with an ECKD device.

extent. Continuous space on a disk or diskette occupied by or reserved for a particular file or VSAM data space.

FASTCOPY, FCOPY. Refer to VSE/Fast Copy

FBA (Fixed Block Architecture) disk device. A disk device that stores data in blocks of fixed size. These blocks are addressed by block number relative to the beginning of the file. Contrast with *CKD device*.

FCB. Forms control buffer.

*** file**. A named set of records stored or processed as a unit. (T) Synonymous with *data set*.

*** forms control buffer (FCB)**. In the 3800 Printing Subsystem, a buffer for controlling the vertical format of printed output.

*** fragmentation (of storage)**. In virtual system, inability to assign real storage locations to virtual storage addresses because the available spaces are smaller than the page size.

general-purpose register. A register, usually explicitly addressable, within a set of registers that can be used for different purposes.

*** generate**. To produce a computer program by selecting subsets of skeletal code under the control of parameters. (A)

generation. See *macro generation*.

*** GETVIS space**. Storage space within a partition or the shared virtual area, available for dynamic allocation to programs.

hard wait. The condition of a processor when all operations are suspended. System recovery from a hard wait is impossible without performing a new system startup.

*** hardware**. All or part of the physical components of an information processing system, such as computers or peripheral devices. (T) (A) Contrast with *software*.

ICCF. See *VSE/ICCF*.

integrated console. In VSE/ESA, the service processor console available on ES/9000 processors that operates as the VSE/ESA system console. The integrated console is typically used during IPL and for recovery purposes when no other console is available.

interactive. A characteristic of a program or system that alternately accepts input and then responds. An interactive system is conversational, that is, a continuous dialog exists between user and system. Contrast with *batch*.

interactive partition. An area of virtual storage for the purpose of processing a job that was submitted interactively via VSE/ICCF.

interface. A shared boundary between two hardware or software units, defined by common functional or physical characteristics. It might be a hardware component or a portion of storage or registers accessed by several computer programs.

*** irrecoverable error**. An error for which recovery is impossible without use of recovery techniques external to the computer program or run. (T)

ISAM interface program. A set of routines that allow a processing program coded to use ISAM to gain access to a VSAM key-sequenced file.

job control statement. A particular statement of JCL.

job step. One of a group of related programs complete with the JCL statements necessary for a particular run. Every job step is identified in the job stream by an EXEC statement under one JOB statement for the whole job.

job stream. The sequence of jobs as submitted to an operating system.

*** librarian**. The set of programs that maintains, services, and organizes the system and private libraries.

library. See *VSE library* and *VSE/ICCF library*.

*** library block**. A block of data stored in a sublibrary.

*** library directory**. The index that enables the system to locate a certain sublibrary of the accessed library.

*** library member**. The smallest unit of data that can be stored in and retrieved from a sublibrary.

*** licensed program**. A separately priced program and its associated materials that bear an IBM copyright and are offered to customers under the terms and conditions of the IBM Customer Agreement (ICA).

*** line printer**. A device that prints a line of characters as a unit. (I) (A) Contrast with *character printer* or *page printer*.

link. 1. To connect items of data or portions of programs; for example, linking of object programs by the linkage editor or linking of data items by pointers. 2. In SNA, the combination of the link connection and the link stations joining network nodes, for example, a System/370 channel and its associated protocols. A link is both logical and physical. Synonymous with *data link*.

*** linkage editor**. A program used to create a phase (executable code) from one ormoren independently translated object modules, from one or more existing phases, or from both. In creating the phase, the linkage editor resolves cross references among the modules and phases available as input. The program can catalog the newly built phases.

linkage stack. An area of protected storage that the system gives to a program to save status information in case of a branch or a program call.

link-edit. To create a loadable computer program by having the linkage editor process compiled (assembled) source programs.

*** local address**. In SNA, an address used in a peripheral node in place of a network address and transformed to or from a network address by the boundary function in a subarea node.

*** lock file**. In a shared disk environment under VSE, a system file on disk used by the sharing systems to control their access to shared data.

logical record. A user record, normally pertaining to a single subject and processed by data management as a unit. Contrast with *physical record* which may be larger or smaller.

logical unit (LU). A name used in programming to represent an I/O device address.

*** logical unit name**. In programming, a name used to represent the address of an input/output unit.

*** main task**. The main program within a partition in a multiprogramming environment.

master console. In VSE/ESA, one or more consoles that receive all system messages, except for those that are directed to one particular console. Contrast this with the *user console* which receives only those messages that are specifically directed to it, for example messages issued from a job that was submitted with the request to echo its messages to that console. The operator of a master console can reply to all outstanding messages and enter all system commands.

*** member**. The smallest unit of data that can be stored in and retrieved from a sublibrary. See also *library member*.

message. 1. In VSE, a communication sent from a program to the operator or user. It can appear on a console, a display terminal or on a printout. 2. In telecommunication, a logical set of data being transmitted from one node to another.

*** module**. A program unit that is discrete and identifiable with respect to compiling, combining with other units, and loading; for example, the input to, or output from an assembler, compiler, linkage editor, or executive routine. (A)

multitasking. Concurrent running of one main task and one or several subtasks in the same partition.

network. 1. An arrangement of nodes (data stations) and connecting branches. 2. The assembly of equipment through which connections are made between data stations.

networking. Making use of the services of a network program.

object module (program). A program unit that is the output of an assembler or compiler and is input to a linkage editor.

online application program. An interactive program used at display stations. When active, it waits for data. Once input arrives, it processes it and sends a response to the display station or to another device.

optical reader/sorter. A device that reads hand written or machine printed symbols on a voucher and, after having read the voucher, can sort it into one of the available stacker-select pockets.

page fault. A program interruption that occurs when a program page marked "not in processor storage" is referred to by an active page.

*** page fixing**. Marking a page so that it is held in processor storage until explicitly released. Until then, it cannot be paged out.

page frame. An area of processor storage that can contain a page.

page-in. The process of transferring a page from the PDS to processor storage.

page-out. The process of transferring a page from processor storage to the PDS.

page printer. A device that prints one page as a unit; for example, a laser printer. Contrast with *character printer, line printer*.

partition. A division of the virtual address area available for running programs. See also *dynamic partition, static partition*.

PASN-AL (primary address space number - access list). The access list that is associated with a partition. A program uses the PASN-AL associated with its partition and the DU-AL associated with its task (work unit). See also *DU-AL*.

Each partition has its own unique PASN-AL. All programs running in this partition can access data spaces through the PASN-AL. Thus a program can create a data space, add an entry for it in the PASN-AL, and obtain the ALET that indexes the entry. By passing the ALET to other programs in the partition, the program can share the data space with other programs running in the same partition.

*** physical record**. The amount of data transferred to or from auxiliary storage. Synonymous with *block*.

primary address space. In VSE/ESA, the address space where a partition is currently executed. A program in primary mode fetches data from the primary address space.

primary mode. If a program runs in primary mode, the system resolves all addresses within the current (primary) address space. Contrast with *AR (access register) mode*. See also *address space control (ASC) mode*.

priority. A rank assigned to a partition or a task that determines its precedence in receiving system resources.

private area. The part of an address space that is available for the allocation of private partitions. Its maximum size can be defined during IPL. Contrast with *shared area*.

*** private partition**. Any of the system's partitions that are not defined as shared. See also *shared partition*.

procedure. See *cataloged procedure*.

*** processing**. The performance of logical operations and calculations on data, including the temporary retention of data in processor storage while this data is being operated upon.

*** processor**. In a computer, a functional unit that interprets and executes instructions. A processor consists of at least an instruction control unit and an arithmetic and logic unit. (T)

processor storage. The storage contained in one or more processors and available for running machine instructions. Synonymous with *real storage*.

*** programmer logical unit**. A logical unit available primarily for user-written programs. See also *logical unit name*.

protocol. In SNA, the set of rules for requests and responses between communicating nodes that want to exchange data.

*** random processing**. The treatment of data without respect to its location on disk storage, and in an arbitrary sequence governed by the input against which it is to be processed.

real address. The address of a location in processor storage.

real mode. A processing mode in which a program may not be paged. Contrast with *virtual mode*.

real storage. See *processor storage*.

record. A set of related data or words, treated as a unit. See *logical record*, *physical record*.

*** reentrant**. The attribute of a program or routine that allows the same copy of the program or routine to be used concurrently by several tasks.

residency mode (RMODE). A program attribute that refers to the location where a program is expected to reside in virtual storage. RMODE 24 indicates that the program must reside in the 24-bit addressable area (below 16 megabytes), RMODE ANY indicates that the program can reside anywhere in 31-bit addressable storage (above or below 16 megabytes).

*** restore**. To write back onto disk data that was previously written from disk onto an intermediate storage medium such as tape.

RMODE. Residency mode.

*** routine**. A program, or part of a program, that may have some general or frequent use. (T)

*** routing**. The assignment of the path by which a message will reach its destination.

SDL. System directory list.

*** search chain**. The order in which chained sublibraries are searched for the retrieval of a certain library member of a specified type.

security. See *access control*.

sequential access. The serial retrieval of records in their entry sequence or serial storage of records with or without a premeditated order. Contrast with *direct access*.

sequential access method (SAM). A data access method that writes to and reads from an I/O device record after record (or block after block). On request, the support performs device control operations such as line spacing or page ejects on a printer or skip a certain number of tape marks on a tape drive.

*** sequential file**. A file in which records are processed in the order in which they are entered and stored.

shared area. An area of storage that is common to all address spaces in the system. VSE/ESA has two shared areas:

1. The shared area (24 bit) is allocated at the start of the address space and contains the supervisor, the SVA (for system programs and the system GETVIS area), and the shared partitions.
2. The shared area (31 bit) is allocated at the end of the address space and contains the SVA (31 bit) for system programs and the system GETVIS area.

*** shared virtual area (SVA)**. A high address area that contains a system directory list (SDL) of frequently used phases, resident programs that can be shared between partitions, and an area for system support.

*** skeleton**. A set of control statements, instructions, or both, that requires user-specific information to be inserted before it can be submitted for processing.

*** software**. All or part of the programs, procedures, rules, and associated documentation of a data

processing system. Software is an intellectual creation that is independent of the medium on which it is recorded. (T)

spanned record. A record that extends over several blocks.

*** spool file**. 1. A file that contains output data saved for later processing. 2. One of three VSE/POWER files on disk: queue file, data file, and account file.

*** spooling**. The use of disk storage as buffer storage to reduce processing delays when transferring data between peripheral equipment and the processors of a computer. In VSE, this is done under the control of VSE/POWER.

*** standard label**. A fixed-format record that identifies a volume of data such as a tape reel or a file that is part of a volume of data.

static partition. A partition, defined at IPL time and occupying a defined amount of virtual storage that remains constant. Contrast with *dynamic partition*.

STOKEN (space token). An eight-byte identifier of a data space. It is generated by the system when you create a data space.

storage fragmentation. Inability to allocate unused sections (fragments) of storage in the real or virtual address range of virtual storage.

*** suballocated file**. A VSE/VSAM file that occupies a portion of an already defined data space. The data space may contain other files. Contrast with *unique file*.

sublibrary. A subdivision of a library. Members can only be accessed in a sublibrary.

sublibrary directory. An index for the system to locate a member in the accessed sublibrary.

submit. A VSE/POWER function that passes a job to the system for processing.

*** subsystem**. A secondary or subordinate system, usually capable of operating independently of, or asynchronously with, a controlling system. (T)

subtask. A task that is initiated by the main task or by another subtask.

*** supervisor**. The part of a control program that coordinates the use of resources and maintains the flow of processor operations.

SYSRES. System residence file.

*** system console**. A console, usually equipped with a keyboard and display screen for control and communication with the system.

system directory list (SDL). A list containing directory entries of frequently-used phases and of all phases resident in the SVA. The list resides in the SVA.

*** system file**. A file used by the operating system, for example, the hardcopy file, the recorder file, the page data set.

system logical unit. A logical unit available primarily for operating system use. See also *logical unit name*.

system residence file (SYSRES). The system sublibrary IJSYSRS.SYSLIB that contains the operating system. It is stored on the system residence volume DOSRES.

system sublibrary. The sublibrary that contains the operating system. It is stored on the system residence volume (DOSRES).

*** tailor**. A process that defines or modifies the characteristics of the system.

*** telecommunication**. Transmission of data between computer systems over telecommunication lines and between a computer system and remote devices.

*** terminal**. A point in a system or network at which data can either enter or leave. (A) Usually a display screen with a keyboard.

*** throughput**. 1. A measure of the amount of work performed by a computer system over a given period of time, for example, number of jobs per day. (I) (A) 2. In data communication, the total traffic between stations per unit of time.

*** track hold**. A function that protects a track that is being updated by one program from being accessed by another program.

*** transient area**. An area within the control program used to provide high-priority system services on demand.

UCS. Universal character set.

*** unique file**. A VSE/VSAM file that occupies a data space of its own. The data space is defined at the same time as the file and cannot contain any other file. Contrast with *suballocated file*.

*** universal character set (UCS)**. A printer feature that permits the use of a variety of character arrays.

user console. In VSE/ESA, a console that receives only those system messages that are specifically

directed to it. These are, for example, messages that are issued from a job that was submitted with the request to echo its messages to that console. Contrast with *master console*.

*** user exit**. A programming service provided by an IBM software product that may be requested during the execution of an application program for the service of transferring control back to the application program upon the later occurrence of a user-specified event.

virtual address. An address that refers to a location in virtual storage. It is translated by the system to a processor storage address when the information stored at the virtual address is to be used.

virtual address area. The virtual range of available program addresses.

virtual disk. A range of up to two gigabytes of contiguous virtual storage addresses that a program can use as workspace. Although the virtual disk exists in storage, it appears as a real FBA disk device to the user program. All I/O operations directed to a virtual disk are intercepted and the data to be written to, or read from, the disk is moved to or from a data space.

Like a data space, a virtual disk can hold only user data; it does not contain shared areas, system data or programs. Unlike an address space or a data space, data is not directly addressable on a virtual disk. To manipulate data on a virtual disk, the program has to perform I/O operations.

Virtual Machine/Enterprise Systems Architecture (VM/ESA). The most advanced VM system currently available.

*** virtual mode**. The operating mode of a program which may be paged.

*** virtual partition**. A division of the dynamic area of virtual storage.

virtual storage. Addressable space image for the user from which instructions and data are mapped into processor (real) storage locations.

VM/ESA. Virtual Machine/Enterprise Systems Architecture.

volume. A data carrier that is mounted and demounted as a unit, for example, a reel of tape or a disk pack. (I) Some disk units have no demountable packs. In that case, a volume is the portion available to one read/write mechanism.

volume ID. The volume serial number, which is a number in a volume label assigned when a volume is prepared for use by the system.

VSE (Virtual Storage Extended). A system that consists of a basic operating system and any IBM supplied and user-written programs required to meet the data processing needs of a user. VSE and the hardware it controls form a complete computing system. Its current version is called VSE/ESA.

VSE/ESA (VSE/Enterprise Systems Architecture). The most advanced VSE system currently available.

VSE/Fast Copy. A utility program for fast copy data operations from disk to disk and dump/restore operations via an intermediate dump file on magnetic tape or disk.

*** VSE/ICCF (VSE/Interactive Computing and Control Facility)**. An IBM program that serves as interface, on a time-slice basis, to authorized users of terminals linked to the system's processor.

VSE/ICCF library. A file composed of smaller files (libraries) including system and user data which can be accessed under the control of VSE/ICCF.

VSE library. A collection of programs in various forms and storage dumps stored on disk. The form of a program is indicated by its member type such as source code, object module, phase, or procedure. A VSE library consists of at least one sublibrary which can contain any type of member.

*** VSE/POWER**. An IBM program primarily used to spool input and output. The program's networking functions enable a VSE system to exchange files with or run jobs on another remote processor.

VSE/VSAM (VSE/Virtual Storage Access Method). An IBM access method for direct or sequential processing of fixed and variable length records on disk devices.

*** VSE/VSAM managed space**. A user-defined space on disk placed under the control of VSE/VSAM.

*** VTAM application program**. A program that has opened an ACB to identify itself to VTAM and can now issue VTAM macro instructions.

wait state. The condition of a processor when all operations are suspended. System recovery from a hard wait is impossible without performing a new system startup. Synonym for *hard wait*.

31-bit addressing. Provides addressability for address spaces of up to 2 gigabytes. (The maximum amount of addressable storage in previous systems was 16 megabytes.)

Index

Numerics

1403
  See printer
24-bit addressing mode 2
2400 series tape
  See tape file, magnetic
2520
  See card I/O
2540
  See card I/O
31-bit addressing
  restrictions 465
  VSE/ESA macros and their mode dependencies 465
31-bit addressing mode 2
3203
  See printer
3211
  See printer
3350
  See disk device
3375
  See disk device
3380
  See disk device
3400 series tape
  See tape file, magnetic
3494 Tape Library, controlling via LBSERV macro 234
3505
  See card I/O
3525
  See card I/O
3540
  See diskette-I/O file
3800
  See printer
3881
  See card I/O
3886
  See MICR/OCR device
4248
  See printer
5203
  See printer
8809
  See tape file, magnetic

A

AB exit routine
  return from 187
abnormal end
  See AB exit routine
abnormal end exit 384
ABSAVE operand (ATTACH macro) 26
access list
  adding an entry (ALESERV ADD) 12
  deleting an entry (ALESERV DELETE) 14
  finding a STOKEN (ALESERV EXTRACT) 15
  searching for an entry (ALESERV SEARCH) 17
ACCESS operand (XECBTAB) 416
access register (AR) mode 2
access-controlled GETVIS subpool 220
activate
  partition (TPOUT) 393
  task (ATTACH) 24
ADDAREA operand (DTFMR) 145
address delimiter 91
ADDRESS operand
  DTFMR macro 145
  MRMOD macro 320
address space
  setting and testing control mode (SYSSTATE macro) 391
address space control (ASC) mode 2
address specification
  dynamically requested storage 218
  FREEVIS macro 200
  GENDTL 202
  I/O request block (GENIORB) 205
  job communication area (JOBCOM) 231
  local directory list 206
  lock file, for modification 317
  lock file, generate dynamically 202
  MODDTL 317
  PDUMP macro 328
  storage, dynamically requested 218
  virtual storage requested 218
address translation
  real address return (REALAD) 350
  virtual address return 396
addressing mode 2
  querying (AMODESW QRY) 20
  returning from subroutine (AMODESW RETURN) 20
  setting (AMODESW SET) 21
  switching (AMODESW CALL) 19
AFTER operand (DTFDA) 117
ALESERV ADD macro 12
ALESERV DELETE macro 14
ALESERV EXTRACT macro 15
ALESERV SEARCH macro 17

© Copyright IBM Corp. 1990, 1999

ALET 11
AMODE 24 2
AMODE 31 2
AMODE operand (STXIT) 388
AMODESW CALL macro 19
AMODESW QRY macro 20
AMODESW RETURN macro 20
AMODESW SET macro 21
AR (access register) mode 2
ASA option control characters 429
ASC (address space control) mode 2
ASCII character code 433
ASCII character set 433
ASCII file
  block prefix 148
  file (DTFMT) 147
  file (DTFPH) 160
  file label for 451
  volume label for 450
ASCII to EBCDIC correspondence 434
ASPL macro 22
ASPL operand 23
assign
  device (ASSIGN) 23
  parameter list (ASPL) 22
associated file
  cardI/O definition (of DTFCD) 108
  print definition (of DTFPR) 164
  print module (PRMOD) 336
  print output 164
associated file (ASOCFLE) operand
  card I/O 108
  DTFCD macro 108
  DTFPF macro 164
  print output 164
attach (a task) 24
AVRLIST macro 27, 215

B

backspace tape 77
backward read (from tape) 151
batch and serial number (BCHSER) operand (DFR) 85
batch number (BCH) operand (DFR) 85
BINARY operand (GETIME) 208
block length
  See also block size (BLKSIZE) operand
  1287/1288 input 154
  card I/O 109
  combined file output 113
  console I/O 115
  direct access file 117
  disk sequential I/O 169
  document-read (3886) file 127
  indexed-sequential file 142
  print output 164
block length *(continued)*
  tape file 147
block size (BLKSIZE) operand
  1287/1288 input 154
  3886 input 127
  card I/O 109
  console I/O 115
  direct access file 117
  disk sequential I/O 169
  DTFCD macro 109
  DTFCN macro 115
  DTFDA macro 117
  DTFDR macro 127
  DTFMT macro 147
  DTFOR macro 154
  DTFPR macro 164
  DTFSD macro 169
  print output 164
  tape I/O 147
blocking factor (BLKFAC) operand
  1287/1288 input 154
  1287/1288 input module 324
  DTFOR macro 154
  ORMOD macro 324
buffer
  additional, MICR file 145
  indexed sequential add 141
  MICR document 70
  MICR/OCR input (DTFMR) 145
  print, clear the 78
  status (MICR) 70
  synchronize, tape 77
buffer offset (BUFOFF) operand (DTFMT) 148
BUFFERS operand
  DTFMR macro 145
  MRMOD macro 320

C

CALL
  CSRPACT 38
  CSRPBLD 32
  CSRPCON 36
  CSRPDAC 39
  CSRPDIS 41
  CSRPEXP 34
  CSRPFRE 45
  CSRPGET 42
  CSRPQCL 53
  CSRPQEX 51
  CSRPQPL 49
  CSRPRFR 47
  CSRPRGT 44
  CSRPxxx 30
call (a program) 28

calling a subroutine (AMODESW CALL macro) 19
cancel program
DUMP macro 180
JDUMP macro 230
STXIT option 386
task 55
card error (CRDERR) operand
card I/O 110
CDMOD macro 65
DTFCD macro 110
card I/O
define I/O module for 65
device control 79
file, define a (DTFCD) 108
carriage spacing, printer 77
CCB
See command control block (CCB)
CCW address (CCWADDR) operand (DTFPH) 160
CCW operand
GENIORB macro 204
IORB macro 224
CDDELETE macro 62
CDLOAD macro 63
CDMOD macro 65
chaining sublibraries 262
CHANGE operand (MODDTL) 318
channel program
command control block for 56
request to execute (EXCP) 186
CHAP macro 68
character
table reference 126, 168, 338
character delimiter 91
character set options (3886 file) 86
character-set (CHRSET) operand (DFR) 86
characters, control 429
CHECK macro 69
checkpoint
end address 73
file on disk 160
on tape, interspersed 148
request macro (CHKPT) 72
restart address 73
CISIZE specification
See control-interval size
CKPTREC operand (DTFMT) 148
CLOCK operand (GETIME) 209
close a file (CLOSE, CLOSER) macro 75
close library member (LIBRM CLOSE) macro 256
CMDCHN operand (DTFDU) 130
code, ASCII character 433
codes, control character 429
combined file
card I/O (DTFCD) 108
console (DTFCN) 115
read from (GET) 207
combined file *(continued)*
write to (PUT) 341
command chaining, diskette I/O 130
command control block (CCB)
address of in EXCP 186
conditions, testable 58
define a (CCB) 56
format of 57
transmission information 58
command list
CCB macro 56
in IORB (CCW operand) 204, 224
command symbols 5
comment specification 5
communication
See also cross-partition communication
activate partition (TPOUT) 393
deactivate partition (TPIN) 393
immediate response request 393
job (step) to job (step) 231
operator interrupt 385
region access (COMRG) macro 79
region, move data to 321
composed page data stream (CPDS) records 164
compressing data 81
compression/expansion macro (CSRCMPSC) 81
connected message 407
console file
combined, write to (PUT) 343
define a (DTFCN) 115
output, write to (PUT) 341
read from (GET) 207
console ID (WTO/R) 407
console message, deleting 93
console messages, issuing 403
console name (WTO/R) 407
control
See also return control
See also virtual storage control
a resource (LOCK) 299
card device 65
character (CDMOD) 65
characters 429
code, I/O command 77
device operation (CNTRL) macro 76
overflow of print page 340
page overflow 167, 337
pocket lights 294
print output 165, 336
printer data check 168
protected resource 177
protected resource, dynamic setting 202
protected resource, modify 318
return of 356
spanned-record file 77
unit, secondary (MICR input) 146

control block
See control information
control character (CTLCHR) operand
ASA option 429
card I/O 110
CDMOD macro 65
DTFCD macro 110
DTFPR macro 165
EBCDIC option 430
print output 165
print output module 336
PRMOD macro 336
control characters
define for print output 165, 336
control information
device (unit), DSECT for 222
DTFPH table 163
look-up (EXTRACT macro) request 189
partition boundary DSECT 302, 303
partition boundary format 302, 303, 307, 308, 312
save area layout DSECT 309
system layout DSECT 312
XTNTXIT information 163
control line (WTO/R) 404
CONTROL operand
1287/1288 input 155
1287/1288 input module 324
card I/O 110
CDMOD macro 65
direct access file 117
DTFCD macro 110
DTFDA macro 117
DTFOR macro 155
DTFPR macro 165
DTL macro 177
GENDTL macro 202
MODDTL macro 318
ORMOD macro 324
print output 165
print output module 336
PRMOD macro 336
resource lock/unlock 177
resource, lock/unlock, dynamic setting 202
resource, lock/unlock, modify 318
control-address specification (CHECK) 69
control-interval size
device independent I/O 123
DTFDI macro 123
DTFPH macro 160
DTFSD macro 170
physical IOCS 160, 161
sequential file on FBA 170
conventions, command 5
conventions, notational 5
CORDATA operand (ISMOD) 226
CORINDX operand (ISMOD) 226
correction exit (COREXIT) operand
1287/1288 input 155
3886 input 127
DTFDR macro 127
DTFOR macro 155
CPCLOSE macro 80
CPDS records 164
cross-partition communication
control block (XPCCB) 421
event control block (XECBTAB) 414
example for use 426
MAPXPCCB macro 313
post event (XPOST) 423
wait on event (XWAIT) 424
XPCC macro 417
XPCCB macro 421
CSRCMPSC macro 81
CSRPACT operand (CALL macro) 38
CSRPBLD operand (CALL macro) 32
CSRPCON operand (CALL macro) 36
CSRPDAC operand (CALL macro) 39
CSRPDIS operand (CALL macro) 41
CSRPEXP operand (CALL macro) 34
CSRPFRE operand (CALL macro) 45
CSRPGET operand (CALL macro) 42
CSRPQCL operand (CALL macro) 53
CSRPQEX operand (CALL macro) 51
CSRPQPL operand (CALL macro) 49
CSRPRFR operand (CALL macro) 47
CSRPRGT operand (CALL macro) 44
CSRPxxx operand (CALL macro) 30
CSRYCMPS macro 82
cylinder overflow (CYLOFL) operand (DTFIS) 135

D

data block
See block length
data check control 168
data compression 81
data expansion 81
data set 2 labels (for MVS) 450
data space
creating (DSPSERV CREATE) 97
deleting (DSPSERV DELETE) 102
dumping (SDUMP/SDUMPX macro) 357
extending (DSPSERV EXTEND) 104
naming conventions 101
releasing (DSPSERV RELEASE) 106
SDUMP/SDUMPX macro 357
data-check control, printer 78
DCTENTRY macro 82, 215
DDKR operand (SECTVAL) 362
DE operand
FETCH macro 196

DE operand *(continued)*
GENL macro 206
LOAD macro 296
deactivate
partition (TPIN) 393
task 84
define
3886 document-read file (DTFDR) 127
card I/O file (DTFCD) 108
console I/O file (DTFCN) 115
device-independent file (DTFDI) 123
direct access file (DTFDA) 117
diskette-unit file (DTFDU) 130
file for physical IOCS (DTFPH) macro 160
indexed-sequential file (DTFIS) 135
MICR/OCR file (DTFMR) 145
OCR file (DTFOR) 154
printer file (DTFPR) macro 164
sequential disk file (DTFSD) macro 169
tape file (DTFMT) 147
VSAM managed sequential 169
define a file
See file definition
define sublibrary chain (LIBRM LIBDEF) macro 262
delete library member (LIBRM DELETE) macro 257
delete loaded phase (CDDELETE) macro 62
DELETFL operand (DTFSD) 170
deleting operator message (DOM macro) 93
dequeue resource (DEQ) macro 83
descriptor code (WTO/R) 406
detach a task (DETACH) macro 84
detail line (WTO/R) 404
device
control (CNTRL) 76
device address (DEVADDR) operand
1287/1288 input 157
3886 input 128
card I/O 110
console I/O 115
device independent I/O 123
direct access file 118
disk sequential I/O 171
diskette I/O 130
DTFCD macro 110
DTFCN macro 115
DTFDA macro 118
DTFDI macro 123
DTFDR macro 128
DTFDU macro 130
DTFMR macro 145
DTFMT macro 148
DTFOR macro 157
DTFPH macro 161
DTFPR macro 166
DTFSD macro 171
MICR/OCR input 145
device address (DEVADDR) operand *(continued)*
physical IOCS 161
print output 166
tape I/O 148
device independence
define I/O module for 87
file for, define a (DTFDI) 123
DEVICE operand
1287/1288 input 157
1287/1288 input module 324
card I/O 110
CDMOD macro 66
diskette I/O 131
DRMOD macro 94
DTFCD macro 110
DTFDR macro 128
DTFDU macro 131
DTFIS macro 135
DTFOR macro 157
DTFPH macro 161
DTFPR macro 166
GENIORB macro 204
I/O request block 204, 224
indexed sequential file 135
IORB macro 224
ORMOD macro 324
physical IOCS 161
print output 166
print output module 336
PRMOD macro 336
DIMOD macro 87
direct access file
define a (DTFDA) 117
read from (READ) 349
trailer labels for 121
wait for end of I/O 399
write to (WRITE) 401
directory entry
mapping 304
directory list, local
See local directory list
directory search only (TXT of LOAD) 297
disengage document feed (DISEN) macro 89
disk area, free a (FREE) 198
disk device
device-independent file on (DTFDI) 123
direct access file on (DTFDA) 117
dynamic assign of 23
indexed-sequential file on (DTFIS) 135
physical IOCS file on (DTFPH) 160
restart address (CHKPT) 74
seek address operation 77
sequential file on (DTFSD) 169
disk file
device independent (DTFDI) 123
direct access (DTFDA) 117

disk file *(continued)*
force end of volume (FEOVD) 194
indexed sequential (DTFIS) 135
physical IOCS (DTFPH) 160
release block (RELSE) 354
sequential (DTFSD) 169
VSAM managed sequential 169
disk file label
continuation 445
first (only) 443
format-1 443
format-3 445
format-4 445
standard 442
user-standard 447
VTOC label 445
diskette label
file, standard 448
volume 448
diskette-I/O file
define I/O module for 179
file, define a (DTFDU) 130
display document field (DSPLY) macro 95
DLINT macro 90
document
buffer 70
feed, stop of (DISEN) 89
field, display of (DSPLY) 95
read (3886) file (DTFDR) 127
read/sort file (DTFMR) 145, 154
DOM macro 93
DRMOD macro 94
drop sublibrary chain (LIBRM LIBDROP) macro 264
DSECT operand (IORB) 225
DSKXTNT operand
direct access file 118
DTFDA macro 118
DTFIS macro 135
indexed sequential file 135
DSPSERV CREATE macro 97
DSPSERV DELETE macro 102
DSPSERV EXTEND macro 104
DSPSERV macro
CREATE request 97
DELETE request 102
EXTEND request 104
RELEASE request 106
DSPSERV RELEASE macro 106
DTFCD macro 108
DTFCN macro 115
DTFDA macro 117
DTFDI macro 123
DTFDR macro 127
DTFDU macro 130
DTFIS macro 135
DTFMR macro 145
DTFMT macro 147
DTFOR macro 154
DTFPH macro 160
DTFPR macro 164
DTFSD macro 169
DTL (define the lock) macro 177
DTL (lock control block)
See lock control block
dual address adapter
definition for 145
module definition for 320
DUMODFx macro 179
DUMP macro 180
dump request by
CANCEL macro 55
DUMP macro 180
JDUMP macro 230
PDUMP macro 328
SDUMP/SDUMPX macro 357
DVCTYP operand (SECTVAL) 363
dynamic generation
directory list 205
I/0 request block (IORB) 204
lock control block 201
dynamic space GETVIS area 218
freeing space from 200
protection of 220

E

EBCDIC option control characters 430
EBCDIC to ASCII correspondence 434
ECB operand
ATTACH macro 26
GENIORB macro 205
IORB macro 225
PAGEIN macro 327
ECHO option (VSE/POWER JECL) 405
edit character (EDCHAR) operand (DFR) 86
EDITn operand (DLINT) 91
end address (CHKPT) 73
end file load mode (ENDFL) macro 182
end line (WTO/R) 404
end of volume
disk, force end of (FEOVD) 194
force, DTFSD 172
records, handling by DTFDA 118
SYSLST/SYSPCH on tape (SEOV) 364
tape, force end of (FEOV) 193
end-of-extent exit routine
physical I/O 161
sequential I/O 171
end-of-file exit routine
1287/1288 input 157
3886 input 128

end-of-file exit routine *(continued)*
card I/O 110
device independent I/O 123
diskette I/O 131
tape I/O 148
end-of-job (EOJ) macro 184
end-set-limit (ESETL) 185
enqueue resource (ENQ) macro 183
EOFADDR operand
1287/1288 input 157
3886 input 128
card I/O 110
device independent I/O 123
disk sequential I/O 171
diskette I/O 131
DTFCD macro 110
DTFDI macro 123
DTFDR macro 128
DTFDU macro 131
DTFMT macro 148
DTFOR macro 157
DTFSD macro 171
tape I/O 148
EOXPTR operand
DTFPH macro 161
DTFSD macro 171
physical I/O 161
sequential I/O 171
erase gap 77
ERASE operand (DFR) 86
erase-character recognition 86
ERRBYTE operand (DTFDA) 118
ERREXT operand
direct access file 118
diskette I/O 131, 179
DTFDA macro 118
DTFDU macro 131
DTFIS macro 136
DTFMT macro 149
DUMODFx macro 179
indexed sequential file 136
indexed sequential I/O module 226
ISMOD macro 226
tape I/O 149
ERROPT operand
See error option
error exit routine
1287/1288 input 155
3886 input 127
card I/O 111
device independent I/O 124
diskette I/O 131
IORB generation 205
print output 166
return from 184
sequential disk I/O 172
error exit routine *(continued)*
tape I/O 149
error option
card I/O 111
device independent I/O 124
disk sequential I/O 171
diskette I/O 131
diskette I/O module 179
DTFCD macro 111
DTFDI macro 124
DTFDU macro 131
DTFMR macro 145
DTFMT macro 149
DTFPR macro 166
DTFSD macro 171
DUMODFx macro 179
MICR/OCR input 145
print output 166
print output module 337
PRMOD macro 337
resource-lock failure 299
tape I/O 149
error retry
card I/O (CRDERR of CDMOD) 65
card I/O (CRDERR of DTFCD) 110
error-return (ERET) macro 184
ESETL macro 185
event control block
See also ECB operand
format of 26
IORB controlled I/O 205, 225
post an (POST) 335
timer 392
execute form
of WTO macro 407
of WTOR macro 412
execute-channel-program (EXCP) macro 186
exit indicator (EXITIND) operand (DTFDR) 128
exit routine
See also exit routine linkage
abnormal end 384
interval timer interrupt 385
MICR/OCR stacker select 145
operator communication interrupt 385
page fault (SETPFA) 369
page overflow 340
program check interrupt 386
return control (EXIT) 187
return from 356
STXIT defined 383
exit routine linkage
abnormal end 384
interval timer interrupt 385
operator communication interrupt 385
program check interrupt 386
set up (STXIT) 383

expanding data 81
EXTADDR operand (DTFMR) 145
extended save area 310
extent checking, direct access file 121
extent checking, physical IOCS 163
extent information 122
extract control information
  macro (EXTRACT) 189
  partition boundary DSECT 302, 303
  save area layout DSECT 309
  system layout DSECT 312
  unit information DSECT (IJBPUB) 222

F

FAIL operand (LOCK) 299
FEED operand (DTFDU) 133
FEOVD operand
  See force end of volume
FETCH (a phase) macro 195
FILABL operand (DTFMT) 150
file
  See also file definition
  3881 (DTFCD) 108
  ASCII, logical IOCS access (DTFMT) 147
  ASCII, physical IOCS access (DTFPH) 160
  associated (DTFCD) 108
  associated, card I/O (DTFCD) 108
  associated, card print (DTFPR) 164
  card I/O (DTFCD) 108
  close a 75
  combined (DTFCD) 108
  combined (DTFCN) 115
  console (DTFCN) 115
  console, write to and read (PUTR) 343
  device independent (DTFDI) 123
  direct access (DTFDA) 117
  diskette I/O (DTFDU) 130
  diskette I/O module 179
  document read (1287/1288) (DTFOR) 154
  document read/sort (DTFMR) 145
  document-read (3886 − DTFDR) 127
  indexed sequential I/O module 226
  indexed sequential, load a (ENDFL) 182
  indexed sequential, load mode (SETFL) 365
  indexed-sequential (DTFIS) 135
  MICR/OCR I/O module (MRMOD) 320
  open a 323
  physical IOCS (DTFPH) 160
  printer (DTFPR) 164
  read from (READ) 349
  read from, sequential (GET) 207
  sequential disk (DTFSD) 169
  tape (DTFMT) 147
  write to (WRITE) 401
  write to, sequential (PUT) 341
file definition
  3881 file (DTFCD) 108
  ASCII, logical IOCS access (DTFMT) 147
  ASCII, physical IOCS access (DTFPH) 160
  associated, card print (DTFPR) 164
  associated, print module (PRMOD) 336
  combined (DTFCN) 115
  console (DTFCN) 115
  device independent (DTFDI) 123
  direct access (DTFDA) 117
  diskette I/O (DTFDU) 130
  document read (1287/1288) (DTFOR) 154
  document read/sort (DTFMR) 145
  document-read (3886 − DTFDR) 127
  indexed-sequential (DTFIS) 135
  physical IOCS (DTFPH) 160
  printer (DTFPR) 164
  sequential disk (DTFSD) 169
  tape (DTFMT) 147
file position control
  note record address (NOTE) 322
  point behind noted record (POINTW) 334
  point to noted record (POINTR) 333
  point to start of file (POINTS) 333
file-security (FILESEC) operand (DTFDU) 133
file-type specification
  See type of file (TYPEFLE) operand
filename specification
  CHECK macro 69
  CHKPT macro 74
  CLOSE (CLOSER) macro 75, 323
  CNTRL macro 76
  DISEN macro 89
  DSPLY macro 95
fix a page (PFIX) macro 329
FLDn operand (DLINT) 90
floating-point registers 4
FONT operand (DFR) 85
force end of volume
  direct access file 118
  disk (FEOVD) 194
  disk sequential I/O 172
  operand, DTFDA macro 118
  operand, DTFSD macro 172
  tape (FEOV) 193
force page out (FCEPGOUT) macro 191
format end (FREND) specification (DLINT) 92
format record (3886 file)
  change of (SETDEV) 364
  define a (DFR) 85
  length of 129
  line type of 90
  phase name of 129
format record, change of (SETDEV) 364
forms control buffer
  load message for 250

forms control buffer *(continued)*
load request (LFCB) macro 249
FORMS operand (LFCB) 250
forward read (from tape) 151
forward space tape 77
free
disk area (FREE) 198
page (PFREE) 331
resource (DEQ) 83
virtual storage (FREEVIS) 199
FREND operand (DLINT) 92
FRNAME operand (DTFDR) 129
function operand
associated (print output) file (FUNC) 167, 337
card I/O (FUNC) 111
CDMOD macro (FUNC) 66
DTFCD macro (FUNC) 111
DTFIS macro (IOROUT) 140
DTFPR macro (FUNC) 167
indexed sequential file 140
indexed sequential I/O module 227
ISMOD macro 227
job-communication 231
PRMOD macro (FUNC) 337
UPDATE (DTFSD) 175

G

gap, erase a 77
GENDTL macro 201
generate I/O request block (GENIORB) macro 204
generation, dynamic
See dynamic generation
GENL macro 205
get library member (LIBRM GET) macro 259
GET macro 207
get symbolic parameter (GETSYMB) macro 210
get time of day (GETIME) macro 208
get virtual storage
See virtual storage request
get virtual storage (GETVIS) macro 217
get volume characteristics (GETVCE) macro 211
GETSYMB macro 210
GETVCE macro 211
GETVCE mapping macros
AVRLIST 27, 82
DCTENTRY 27, 82
GETVIS area
dynamic space 220
location of 218
partition 218
SVA 221
GETVIS macro 217
GETVIS subpool, access-controlled 220
Greenwich mean time 209

H

HDRINFO operand
DTFMT macro 150
DTFPH macro 161
physical IOCS 161
tape I/O 150
header information
print for physical IOCS 161
print for tape I/O 150
HEADER operand
1287/1288 input 158
3886 input 129
DTFDR macro 129
DTFOR macro 158
high index (HINDEX) operand (DTFIS) 138
HOLD operand
direct access file 118
disk sequential I/O 172
DTFDA macro 118
DTFIS macro 138
DTFSD macro 172
indexed sequential file 138
indexed sequential I/O module 227
ISMOD macro 227
hopper empty (1287D, 1288) 158
horizontal copy control 78
HPRMTY operand (DTFOR) 158

I

I/O request block (IORB)
address of in EXCP 186
assembly of, in program 224
define dynamically 204
I/O request block (IORB) macro 224
I/O-area size
See block size (BLKSIZE) operand
I/O-area specification
1287/1288 input 158
1287/1288 input module 324
additional (DTFIS) 140
card I/O 112
CDMOD macro 66
console file 115
DIMOD macro 87
direct access file 119
disk sequential I/O 172
diskette I/O 133
DTFCD macro 112
DTFDA macro 119
DTFDI macro 124
DTFDR macro 129
DTFDU macro 133
DTFMR macro 146
DTFMT macro 151

I/O-area specification *(continued)*
DTFOR macro 158
DTFPR macro 167
DTFSD macro 172
indexed sequential I/O module 227
ISMOD macro 227
load file (DTFIS) 139
MICR/OCR input 146
ORMOD macro 324
print output 167
print output module 337
PRMOD macro 337
randomly process file (DTFIS) 139
sequentially process file (DTFIS) 140
tape I/O 151
I/O-register specification
1287 input 158
card I/O 112
disk sequential I/O 173
diskette I/O 133
DTFCD macro 112
DTFDI macro 124
DTFDU macro 133
DTFIS macro 140
DTFMR macro 146
DTFMT 152
DTFMT macro 151
DTFOR macro 158
DTFPR macro 167
DTFSD macro 173
indexed sequential file 140
MICR/OCR input 146
print output 167
tape I/O 151
IDLOC operand (DTFDA) 118
IJBPUB (unit information DSECT) macro 222
IJJLBSER macro 222
IMAGE operand (DLINT) 90
index of GETVIS subpool 220
index specification
area (INDAREA of DTFIS) 138
high, unit of (HINDEX of DTFIS) 138
master (MSTIND of DTFIS) 142
size (INDSIZE of DTFIS) 139
skip (INDSKIP of DTFIS) 138
indexed sequential file
group retrieval 367
indexed-sequential I/O DTF 135
module for 226
read from by key (READ) 349
read from, sequential (GET) 207
sequential access, start of 367
set load mode (SETFL) 365
wait for end of I/O 399
write to by key (WRITE) 401
INPSIZE operand (DTFCN) 115
insert records 141
interfaces, programming xiii
interval timer
See also IT exit routine
event control block (TECB) 392
set the (SETIME) 365
interval timer interrupt 385
IOSIZE operand (DTFIS) 141
ISMOD macro 226
IT exit routine
return from 187

J

JDUMP macro 230
job name
CPCLOSE parameter list 80
job-communication (JOBCOM) macro 231
job, end of (EOJ) 184
journal tape processing
blocking line records 154, 324
device specification for 157, 324
online correction 348
record length for 159

K

KEEP operand
DTL macro 177
GENDTL macro 202
MODDTL macro 319
key argument (KEYARG) operand
direct access file 119
DTFDA macro 119
DTFIS macro 141
indexed sequential file 141
key length (KEYLEN) operand
direct access file 119
DTFDA macro 119
DTFIS macro 141
indexed sequential file 141
key location (KEYLOC) operand (DTFIS) 142

L

label address (LABADDR) operand
direct access file (DTFDA) 119
disk sequential I/O (DTFSD) 173
physical IOCS file (DTFPH) 162
tape file (DTFMT) 151
label exit routine
coding a 439
direct access file (DTFDA) 119
disk sequential I/O (DTFSD) 173
physical IOCS file (DTFPH) 162

label exit routine *(continued)*
tape file (DTFMT) 151
label formats
data set 2 labels (for MVS) 450
disk volume 442
disk-user, standard 447
diskette volume 448
diskette, file standard 448
format-1 443
format-3 445
format-4 445
tape, file, standard 450
tape, user-standard file 454
tape, volume 449
label line (WTO/R)
heading 404
label line 404
label-extent information 122
label-identification code 439
labels, user
delete format-1 170
extent checking, direct access file 121
extent checking, physical IOCS 163
header information (on tape) 150, 161
identification code 439
nonstandard, processing of 441
physical-IOCS file 162
processing of 439
return control (LBRET) 232
routine for, direct access file 119
routine for, disk sequential I/O 173
routine for, file on tape 151
tape, nonstandard 454
trailer, for direct-access file 121
type of on tape 150
LBRET macro 232
LBSERV macro 234
LDCB (librarian data control block) 252
showing length of 279
length check (LENCHK) operand (DTFMT) 151
LENGTH operand
FREEVIS macro 200
GENDTL macro 202
GENIORB macro 205
GETVIS macro 218
I/O request block (GENIORB) 205
job communication data 231
JOBCOM macro 231
lock file (GENDTL) 202
virtual storage request 218
level test, subsystem 389
LFCB macro 249
LFR operand (DLINT) 90
librarian data control block (LDCB) 252
showing length of 279
Library Access Service macros
LIBRDCB 252
LIBRM CLOSE 256
LIBRM DELETE 257
LIBRM GET 259
LIBRM LIBDEF 262
LIBRM LIBDROP 264
LIBRM LOCK 265
LIBRM NOTE 266
LIBRM OPEN 269
LIBRM POINT 273
LIBRM PUT 275
LIBRM RENAME 277
LIBRM SHOWCB 279
LIBRM STATE CHAIN 280
LIBRM STATE LIB 283
LIBRM STATE MEMBER 285
LIBRM STATE SUBLIB 289
LIBRM UNLOCK 292
LIBRDCB macro 252
LIBRM CLOSE macro 256
LIBRM DELETE macro 257
LIBRM GET macro 259
LIBRM LIBDEF macro 262
LIBRM LIBDROP macro 264
LIBRM LOCK macro 265
LIBRM NOTE macro 266
LIBRM OPEN macro 269
LIBRM POINT macro 273
LIBRM PUT macro 275
LIBRM RENAME macro 277
LIBRM SHOWCB macro 279
LIBRM STATE CHAIN macro 280
LIBRM STATE LIB macro 283
LIBRM STATE MEMBER macro 285
LIBRM STATE SUBLIB macro 289
LIBRM UNLOCK macro 292
LINBEG operand (DLINT) 90
line format (DLINT) 90
line type (WTO/R) 404
line-group definition (DFR) 85
lines-per-inch setting 250
LIOCS module, define a
3886 file (DRMOD) 94
card I/O (CDMOD) 65
device independent (DIMOD) 87
diskette I/O (DUMODFx) 179
IJMxxxxx (1287/1288 input) 325
indexed sequential file (ISMOD) 226
MICR/OCR file (MRMOD) 320
print output (PRMOD) 336
list form
of WTO macro 407
of WTOR macro 412
LITE macro 294

load
phase, retaining control 295
load forms control buffer (LFCB) macro 249
load mode
end of 182
file definition for 140
load phase (LOAD) macro 295
load request
See phase, load a
LOC operand (GETVIS) 218
local directory list
address of 206
define a (GENL) 205
format definition (in FETCH) 196
format definition (in GENL) 206
format definition (in LOAD) 296
scan of by FETCH 196
scan of by LOAD 296
search of only (TXT of LOAD) 297
local time 209
lock a resource (LOCK) macro 299
lock control
control block for (DTL) 177
control block for (GENDTL) 201
system action summary 300
unlock resource (UNLOCK) 395
lock control block
assembly of, in program 177
define dynamically 201
modify a (MODDTL) 317
lock library member (LIBRM LOCK) macro 265
lock option (LOCKOPT) operand
DTL macro 177
GENDTL macro 202
MODDTL macro 318
logical unit (LOGUNIT) operand
GENIORB macro 205
logical unit (programmer), release of 351
LOGUNIT operand (SECTVAL) 363
lower/upper-case control 78
LPI operand (LFCB) 250

M

macro
coding fields of 4
definition of 1
notation 5
operands 4
register notation in 8
usage 1
macro-format generate (MFG) operand
ATTACH macro 27
FETCH macro 197
GETIME macro 209
LOAD macro 298
macro-format generate (MFG) operand *(continued)*
PDUMP macro 328
STXIT macro 387
SUBSID macro 389
XECBTAB macro 416
macros and their mode dependencies 465
magnetic ink character reader
See MICR/OCR device
MAPBDY macro 302
MAPBDYVR macro 303
MAPDNTRY macro 304
MAPEXTR macro 306
MAPSAVAR macro 309
MAPSSID macro 311
MAPSYSP macro 312
MAPXPCCB macro 313
message (WTO/R)
connected 407
control line 404
deleting (DOM macro) 93
descriptor code 406
detail line 404
line type 404
multiple-line 404
routing 405
routing code 405
single line 404
special handling 407
text 404
title 404
message reply (WTOR) 410
message writing (WTO) 403
MFG operand
See macro-format generate (MFG) operand
MIC operand (GETIME) 208
MICR document buffer 70
MICR/OCR device
1287/1288 file module (ORMOD) 324
1287/1288 file, define a (DTFOR) 154
1419 sort mode 146
3886 file, define a (DTFDR) 127
3886 I/O module, define a 94
character set options (3886 file) 86
document field, display of 95
dual address adapter 145
format/line-group definition (3886 file) 85
line format (3886 file) 90
pocket-lights control 294
read from (READ) 349
selective reread (RESCN) 354
sort mode, 1419 146
wait for end of I/O 399
write to (WRITE) 401
mode dependencies of VSE/ESA macros 465
MODE operand (DTFCD) 112

modify lock control block (MODDTL) macro 317
module name (MODNAME) operand
1287/1288 input 159
3886 input 129
card I/O 113
console I/O 116
diskette I/O 133
DTFCD macro 113
DTFCN macro 116
DTFDI macro 125
DTFDR macro 129
DTFDU macro 133
DTFIS macro 142
DTFMR macro 146
DTFOR macro 159
DTFPR macro 167
IJCxxxxx (card file) 67
IJDxxxxx (print output) 338
IJHxxxxx (indexed-sequential I/O) 228
IJJxxxxx (device independent) 88
IJMxxxxx 325
IJMZxxxx (3886 file) 94
IJNDxxxx (diskette file) 179
IJUxxxxx (MICR/OCR input) 320
indexed sequential file 142
MICR/OCR input 146
print output 167
module, LIOCS, sub/supersetting
3886 file (IJCxxxxx) 94
card I/O (IJCxxxxx) 67
device independent (IJJxxxxx) 88
diskette I/O (IJNDxxxx) 180
IJDxxxxx (print output) 339
IJHxxxxx 229
MOUNTED operand (DTFPH) 162
move to communication region (MVCOM) 321
MRMOD macro 320
MSTIND operand (DTFIS) 142
MVCOM macro 321

N

NAME operand
ATTACH (subtask) macro 26
NATNHP operand (DFR) 86
nonstandard labels, processing of 439
nonstandard user label 454
NOSCAN operand (DLINT) 90
notation
conventions of 5
macros, general 5
operand 9
register 8
notations, command 5
note member address (LIBRM NOTE) macro 266
note record address (NOTE) macro 322
NRECDS operand (DTFIS) 142
NULMSG operand (LFCB) 250
numeric hand printing 86

O

OC exit routine
return from 187
open a file (OPEN, OPENR) macro 323
open library member (LIBRM OPEN) macro 269
operand
keyword 5
positional 5
operand notation 9
operator communication
See OC exit routine
operator verification
address of 74
operator, writing to (WTO) 403
operator, writing to with reply (WTOR) 410
optical character reader
See MICR/OCR device
optical reader module (ORMOD) macro 324
OPTION operand (STXIT) 386
ORMOD macro 324
OUBLKSZ operand (DTFCD) 113
overflow control 340
OWNER operand
DTL macro 178
GENDTL macro 203
MODDTL macro 319

P

page boundary
load at (CDLOAD) 63
page control
force page out 191
page fault appendage (SETPFA) 369
page-fix request (PFIX) 329
page-free request (PFREE) 331
page-in request (PAGEIN) 326
release a page (RELPAG) 352
page fault appendage
coding requirements 370
linkage for (SETPFA) 369
page fault queue in 371
processing by 371
page fault queue, internal 372
PAGE operand (GETVIS) 219
page overflow
See print overflow
page release request (RELPAG) macro 352
page-fix
list 329

page-fix *(continued)*
request (PFIX) macro 329
page-free
list 331
request (PFREE) macro 331
page-in
list 326
request (PAGEIN) macro 326
page-overflow exit 340
page-release list 352
PAGEIN macro 326
parameter list (CP CLOSE) 80
partition
activate (TPOUT) 393
deactivate (TPIN) 393
partition boundary DSECT 302, 303
partition boundary information
format of 302, 303, 307, 308, 312
request return of 189
PC exit routine
return from 187
PDUMP macro 328
PFIX macro 329
PFIX operand (GETVIS) 219
phase-name specification
CDDELETE macro 62
CDLOAD macro 63
phase, load a
into partition GETVIS area (CDLOAD) 63
local directory scan (FETCH) 196
local directory scan (LOAD) 296
passing control (FETCH) 195
retain control unconditionally 298
retaining control (LOAD) 295
system directory scan (FETCH) 196
system directory scan (LOAD) 296
physical IOCS
access request (EXCP) 186
command control block for 56
define file for (DTFPH) 160
disk I/O with RPS 362
I/0 request block (IORB) 204, 224
sector value (SECTVAL) 362
wait for end of I/O (WAIT) 397
physical write on FBA 173
PIOCS
See physical IOCS
pocket-lights control 294
point macros
behind noted record (POINTW) 334
to noted record (POINTR) 333
to start of file (POINTS) 333
point to noted member record (LIBRM POINT) macro 273
POINTRG operand
XPOST macro 423
POINTRG operand *(continued)*
XWAIT macro 425
POOL operand (GETVIS) 219
position-file control
note record address (NOTE) 322
point behind noted record (POINTW) 334
point to noted record (POINTR) 333
point to start of file (POINTS) 333
post ECB (POST) macro 335
precision (PREC) operand (SETIME) 366
primary mode 2
print file (VM) release (CPCLOSE) macro 80
print overflow
operand (PRINTOV of DTFPR) 167
operand (PRINTOV of PRMOD) 337
printer
character, table reference 168, 338
control characters, ASA 429
control of, defining 165, 336
data check control 168
device control 77
file, define a (DTFPR) 164
output module (PRMOD) 336
overflow (PRTOV) macro 340
page overflow on 167, 337
selective tape listing 168, 337
table reference character 168, 338
printer-carriage spacing 77
priority, change for a task 68
private subtasks 369
PRMOD macro 336
program check
See PC exit routine
program check interrupt 386
program, calling a 28
programmer logical unit, release of 351
programming interfaces xiii
PRTOV macro 340
publications, related xv
punch file (VM) release (CPCLOSE) macro 80
put library member (LIBRM PUT) macro 275
PUT macro 341
PUTR macro 343
PWRITE operand (DTFSD) 173

R

random retrieval
See indexed sequential file
RC operand (DUMP) 181
RCB, define a 348
rcbname specification
DEQ macro 83
reactivate partition (TPOUT) 393
read a line (RDLNE) macro 348

READ macro 349
read mode, direct access 119
READ operand (DTFMT) 151
read request
  sequential (GET) 207
read-only module
  3886 input 129
  card I/O 66, 113
  device independent 87
  diskette I/O 133
  diskette I/O module 179
  DRMOD macro 94
  DTFCD macro 113
  DTFDI macro 125
  DTFDR macro 129
  DTFDU macro 133
  DTFIS macro 142
  DTFPR macro 167
  DUMODFx macro 179
  indexed sequential file 142
  indexed sequential I/O module 227
  ISMOD macro 227
  print output 167
  print output module 337
  PRMOD macro 337
READID operand (DTFDA) 119
READKEY operand (DTFDA) 119
real address (in CCW) 186
real address return (REALAD) macro 350
RECFORM operand (SECTVAL) 363
record format (RECFORM) operand
  1287/1288 input 159
  1287/1288 input module 324
  card I/O 113
  console I/O 116
  direct access file 119
  disk sequential I/O 173
  DTFCD 113
  DTFCN macro 116
  DTFDA macro 119
  DTFIS 142
  DTFPR macro 168
  DTFSD macro 173
  indexed-sequential file 142
  indexed-sequential I/O module 227
  ISMOD macro 227
  print output 168
  print output module 337
  PRMOD macro 337
  tape I/O 152
record length
  device-independent file 125
  diskette-I/O file 134
  journal tape processing 159
  MICR/OCR file (DTFMR) 146
  tape I/O 152
record length *(continued)*
  undefined, card I/O 113
  undefined, console I/O 116
  undefined, direct access 120
record length, print output 164
record size (RECSIZE) operand
  1287/1288 input 159
  card I/O 113
  console I/O 116
  direct access file 120
  disk sequential I/O 174
  diskette I/O 134
  DTFCD macro 113
  DTFCN macro 116
  DTFDA macro 120
  DTFDI macro 125
  DTFDU macro 134
  DTFIS macro 142
  DTFMR macro 146
  DTFMT macro 152
  DTFOR macro 159
  DTFPR macro 168
  DTFSD macro 174
  indexed-sequential file 142
  MICR/OCR input 146
  print output 168
  tape I/O 152
records, number of (DTFIS) 142
register
  floating-point 4
  notation 8
  save area 4
  saving of (SAVE) 357
  usage 4
REJECT operand (DFR) 87
related publications xv
release
  block of data (RELSE) 354
  disk area (FREE) 198
  page, virtual storage 352
  programmer logical unit 351
  resource (DEQ) 83
  virtual storage (FREEVIS) 199
release block (RELSE) macro 354
release logical unit (RELEASE) macro 351
release page (RELPAG) macro 352
RELPAG macro 352
RELTYPE operand (DTFDA) 120
rename library member (LIBRM RENAME) macro 277
replying to message (WTOR) 410
RESCN macro 354
resource control
  dequeue an RCB 83
  enqueue RCB (ENQ) 183
  lock a (LOCK) 299
  unlock a (UNLOCK) 395

resource control block
CCB 56
DTFCD 108
DTFCN 115
DTFDA 117
DTFDI 123
DTFDR 127
DTFDU 130
DTFIS 135
DTFMR 145
DTFMT 147
DTFOR 154
DTFPH 160
DTFPR 164
DTFSD 169
event control block (ECB) 26
IORB 224
RCB 348
XECBTAB 414
resource control block (RCB) macro 348
restart
disk address for 74
end address 73
instruction-address for 73
tape position for 73
restart address (CHKPT) 73
RET operand (LOAD) 298
RETPNF operand (CDLOAD) 63
retrieve library member (LIBRM GET) macro 259
retrieve records 141
return control
interrupt exit (EXIT) 187
label processing (LBRET) 232
return control (RETURN) macro 356
returning from a subroutine (AMODESW RETURN macro) 20
rewind tape 77
on OPEN 152
programmed request (CNTRL) 77
rewind tape (REWIND) operand (DTFMT) 152
rotational position sensing 228
rotational position sensing (RPS) 362
routing code (WTO) 405
RPS operand (ISMOD) 228
RUNMODE macro 356

S

save area layout (MAPSAVAR macro) 309, 310
save area, extended 310
save area, subtask 25
SAVE operand
ASSIGN macro 23
ATTACH macro 25
DETACH macro 84
POST macro 335
save register (SAVE) macro 357
SCOPE operand
DTL macro 178
GENDTL macro 203
MODDTL macro 319
SDUMP macro 357
SDUMPX macro 357
search library (LIBRM STATE LIB) macro 283
search library chain (LIBRM STATE CHAIN) macro 280
search library member (LIBRM STATE MEMBER) macro 285
search multiple tracks 121
search sublibrary (LIBRM STATE SUBLIB) macro 289
SECADDR operand (DTFMR) 146
secondary control unit (MICR input) 146
sector value (SECTVAL) macro 362
seek address 77
SEEKADR operand (DTFDA) 120
selective online correction 348
selective reread (RESCN) 354
selective tape list feature 168, 337
selective tape listing 341
sense-CCW specification (CCB) 56
separate assembly
1287/1288 input 159
1287/1288 input module 324
3886 input 130
CDMOD macro 66
device independent 87
disk sequential I/O 174
diskette I/O 134
diskette I/O module 179
DRMOD macro 94
DTFCD macro 114
DTFDA macro 120
DTFDI macro 126
DTFDR macro 130
DTFDU macro 134
DTFIS macro 143
DTFMR macro 146
DTFMT macro 152
DTFOR macro 159
DTFPR macro 168
DTFSD macro 174
DUMODFx macro 179
indexed-sequential file 143
indexed-sequential I/O module 228
ISMOD macro 228
MICR/OCR input 146
MICR/OCR input module 320
MRMOD macro 320
ORMOD macro 324
print output 168
print output module 337
PRMOD macro 337

separate assembly *(continued)*
tape I/O 152
sequential access
end of (ESETL) 185
start of 367
sequential file
1287/1288 input DTF 154
3886 input DTF 127
3886 input module 94
card I/O DTF 108
card I/O module 65
console I/O DTF 115
device independent module 87
device-independent I/O DTF 123
disk I/O DTF 169
diskette I/O DTF 130
diskette I/O module 179
print output DTF 164
read from (GET) 207
release block (RELSE) 354
tape DTF 147
write to (PUT) 341
set and test macro level (SPLEVEL) macro 382
set device (SETDEV)
operand (DRMOD) 94
operand (DTFDR) 130
set exit (STXIT) 383
set file load mode (SETFL) macro 365
set interval timer (SETIME) macro 365
set limits (SETL) macro 367
set page fault appendage (SETPFA) macro 369
SETDEV macro 364
SETPRT macro 373
setting the addressing mode (AMODESW SET macro) 21
short block
See truncation of data
show librarian control block (LIBRM SHOWCB) macro 279
sort mode, 1419 146
SORTMDE operand (DTFMR) 146
spanned records
skipping of (by CNTRL) 77
spacing over 354
SPID operand
FREEVIS macro 200
GETVIS macro 220
SPLEVEL macro 1, 382
spool file (VM) release (CPCLOSE) macro 80
SRCHM operand (DTFDA) 121
SSELECT operand (DTFCD) 114
stacker eject/selection
card I/O 79
card I/O (SSELECT of DTFCD) 114
control characters, ASA 429
control characters, EBCDIC 430
stacker eject/selection *(continued)*
document I/O 79
standard label
IBM file, disk 442
IBM file, diskette 448
IBM file, tape 450
IBM volume, diskette 448
IBM volume, tape 449
processing of 439
user file, disk 446
user file, tape 454
STANDARD operand (GETIME) 208
STLIST operand
DTFPR macro 168
PRMOD macro 337
STLSK operand (PUT) 342
STLSP operand (PUT) 341
storage control
See virtual storage control
storage dump
See dump request by
STXIT routine
interrupt in 388
set up an 383
sublibrary chain, defining a 262
sublibrary chain, dropping a 264
subpool (GETVIS)
access-controlled 220
creating 220
subpool index
FREEVIS 200
GETVIS 220
subroutine
See exit routine
subsystem information
level test 389
mapping DSECT for 311
request (SUBSID) macro 389
subtask
See task
SVA operand
FREEVIS macro 200
GETVIS macro 221
symbolic parameter, getting value of 210
synchronize tape buffer 77
syntax symbols 5
syntax, of commands 5
SYS operand
FETCH macro 196
LOAD macro 296
SYSSTATE macro 2, 391
system directory list
scan of by FETCH 196
scan of by LOAD 296
search of only (TXT of LOAD) 297

system end-of-volume (SEOV) macro 364
system GETVIS area 221
system layout DSECT 312
SYSxxx specification
  CCB macro 56
  CHKPT macro 73
  forms control buffer load device 249
  LFCB macro 249
  RELEASE macro 351

T

table reference character 87, 126, 168, 338
tape buffer, synchronization of 77
tape device
  dynamic assign of 23
  restart position (CHKPT) 73
tape file, magnetic
  define a (DTFMT) 147
  device control 77
  end of volume SYSLST/SYSPCH (SEOV) 364
  force end of (FEOV) 193
  physical IOCS file (DTFPH) 160
  release block (RELSE) 354
  SYSLST/SYSPCH end of volume (SEOV) 364
tape handling (3494) via LBSERV macro 234
tape label
  data set 2 labels (for MVS) 450
  file, standard 450
  file, user-standard 454
  nonstandard 454
  standard, file 450
  standard, volume 449
  volume 449
tape mark
  write a (CNTRL) 77
  write a (DTFMT) 153
task
  attach a (ATTACH) 24
  cancel a 55
  deactivate a (DETACH) 84
  privately controlled 369
  save area for 25
  set low priority for 68
  wait on programmed event 397
task subpool 221
task-to-task communication
  dequeue resource (DEQ) 83
  enqueue resource (ENQ) 183
  event control block for 26
  post ECB (POST) 335
TECB macro 392
telecommunication
  See communication
test interval timer (TTIMER) 394
time of day request (GETIME) macro 208
timer
  interval, event control block (TECB) 392
  interval, setting of (SETIME) 365
  interval, testing the (TTIMER) 394
TPIN macro 393
TPOUT macro 393
track balance, getting (GETVCE) 211
track capacity, getting (GETVCE) 211
trailer labels, direct access file 121
translation, ASCII to EBCDIC to ASCII 433
transmission information, CCB 58
TRC (table reference character) operand
  DTFPR macro 168
  PRMOD macro 338
TRC operand (DIMOD) 87
TRC operand (DTFDI) 126
TRLBL operand (DTFDA) 121
truncate (TRUNC) macro 394
truncation of data
  disk sequential I/O 174
  request macro (TRUNC) 394
  TRUNCS operand, DTFSD macro 174
TSKSUBP operand (GETVIS) 221
TTIMER macro 394
TU operand (GETIME) 208
TXT operand (LOAD) 297
type of file (TYPEFLE) operand
  card I/O 114
  CDMOD macro 66
  console I/O 116
  DIMOD macro 88
  direct access file 121
  disk sequential I/O 174
  diskette I/O 134
  DTFCD macro 114
  DTFCN macro 116
  DTFDA macro 121
  DTFDU macro 134
  DTFIS macro 143
  DTFMT macro 153
  DTFPH macro 163
  DTFSD macro 174
  indexed-sequential file 143
  indexed-sequential I/O module 228
  ISMOD macro 228
  physical IOCS 163
  tape I/O 153
TYPE operand (XECBTAB) 414

U

UCS operand (DTFPR) 168
unit information
  request return of 189

unit record device
dynamic assign of 23
unit-information DSECT (IJBPUB) macro 222
unload tape
on CLOSE 152
programmed request (CNTRL) 77
unlock library member (LIBRM UNLOCK) macro 292
unlock resource (UNLOCK) macro 395
update a file
indexed sequential 141
UPDATE operand (DTFSD) 175
upper/lower-case control 78
use of registers 4
user label
nonstandard, processing of 441
processing of 439
standard, format of 446
user labels
See labels, user
using macros 1

V

variable-length record build (VARBLD) operand
disk sequential I/O 175
DTFMT macro 153
DTFSD macro 175
tape I/O 153
VERIFY operand
direct access file 121
disk sequential I/O 175
diskette I/O 134
DTFDA macro 121
DTFDU macro 134
DTFIS macro 143
DTFSD macro 175
indexed-sequential file 143
virtual address return (VIRTAD) macro 396
virtual storage control
force page out 191
free virtual storage (FREEVIS) 199
get virtual storage (GETVIS) 217
page fault appendage (SETPFA) 369
page-fix request (PFIX) 329
page-free request (PFREE) 331
page-in request (PAGEIN) 326
real address return (REALAD) 350
release a page (RELPAG) 352
run mode notification 356
virtual address return 396
virtual storage release
address of area 200
from dynamic space GETVIS area 200
from subpool 200
from SVA 200
length of area 200
virtual storage release *(continued)*
macro (FREEVIS) 199
virtual storage request
from dynamic space GETVIS area 220
from pool 219
from system GETVIS 221
GETVIS macro 217
location of GETVIS area 218
page boundary allocation 219
PFIXed area 219
subpool for 220
VM spool file release (CPCLOSE) macro 80
VOLSEQ operand (DTFDU) 134
volume characteristics, getting (GETVCE) 211
volume identifier (VOLID) operand
DTL macro 178
GENDTL macro 203
MODDTL macro 319
volume label
disk 442
volume, force end of
disk (FEOVD) 194
SYSLST/SYSPCH on tape (SEOV) 364
tape (FEOV) 193

W

wait state
beginning of for a task (WAIT) 397
ending of for a task 335
file access (WAITF) 399
MICR file read (CHECK) 69
multiple events (WAITM) 400
work file read/write (CHECK) 69
WLRERR operand
device independent I/O 126
disk sequential I/O 175
DTFDI macro 126
DTFMT macro 153
DTFSD macro 175
tape I/O 153
work area specification
1287/1288 input 159
1287/1288 input module 325
CDMOD macro 67
console I/O 116
disk sequential I/O 176
diskette I/O 134
DTFCD macro 114
DTFCN macro 116
DTFDU macro 134
DTFMT macro 153
DTFOR macro 159
DTFPR macro 169
DTFSD macro 176
indexed sequential file 143

work area specification *(continued)*
ORMOD macro 325
print output 169
print output module 338
PRMOD macro 338
read request (GET) 207
tape I/O 153
write request (PUT) 341
write request (PUTR) 343
work file
disk sequential I/O 174
point behind noted record (POINTW) 334
point to noted record (POINTR) 333
point to start of file (POINTS) 333
read from (READ) 349
tape 153
wait for end of I/O (CHECK) 69
WORKL operand (DTFIS) 143
WORKR operand (DTFIS) 143
WORKS operand (DTFIS) 143
WRITE macro 401
write protect (on diskette) 134
write tapemark
on CLOSE (DTFMT) 153
programmed request (CNTRL) 77
write to file
by record ID 121
by record key 121
with formatting 117
write to operator with reply (WTOR) macro 410
write-to-operator (WTO) macro 403
WRITEID operand (DTFDA) 121
WRITEKY operand (DTFDA) 121
writing messages (WTO) 403
wrong-length-record exit routine
device independent I/O 126
disk sequential I/O 175
tape I/O 153
WRTPROT operand (DTFDU) 134
WTO macro 403
execute form 407
list form 407
WTOR macro 410
execute form 412
list form 412

X

X-macro (SDUMPX) 3
XECB operand
XECBTAB macro 415
XPOST macro 423
XWAIT macro 424
XECB table entry
access type 416
check a 414
XECB table entry *(continued)*
define a 414
delete a 415
name of 415
post a (XPOST) 423
reset a 415
wait on (XWAIT) 424
XPCC macro 417
XPCCB (cross-partition communication control block) 421
XPCCB macro 421
XPOST macro 423
XTNTXIT operand (DTFDA) 121
XWAIT macro 424

Y

Year 2000 support
YEAR224 macro 426
YEAR224 macro 426

Communicating Your Comments to IBM

IBM VSE/Enterprise Systems Architecture
System Macros Reference
Version 2 Release 4

Publication No. SC33-6716-00

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM. Whichever method you choose, make sure you send your name, address, and telephone number if you would like a reply.

Feel free to comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of the book. However, the comments you send should pertain to only the information in this manual 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.

If you are mailing a readers' comment form (RCF) from a country other than the United States, you can give the RCF to the local IBM branch office or IBM representative for postage-paid mailing.

- If you prefer to send comments by mail, use the RCF form and either send it postage-paid in the United States, or directly to:

  IBM Deutschland Entwicklung GmbH
  Department 3248
  Schoenaicher Strasse 220
  D-71032 Boeblingen
  Federal Republic of Germany

- If you prefer to send comments by FAX, use this number:
  - (Germany): 07031-16-3456
  - (Other countries): (+49)+7031-16-3456

- If you prefer to send comments electronically, use this network ID:

  IBM Mail Exchange: DEIBMBM9 at IBMMAIL

  INTERNET: s390id@de.ibm.com

Make sure to include the following in your note:

- Title and publication number of this book
- Page number or topic to which your comment applies.

Readers' Comments — We'd Like to Hear from You

IBM VSE/Enterprise Systems Architecture
System Macros Reference
Version 2 Release 4

Publication No. SC33-6716-00

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	□	□	□	□	□

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	□	□	□	□	□
Complete	□	□	□	□	□
Easy to find	□	□	□	□	□
Easy to understand	□	□	□	□	□
Well organized	□	□	□	□	□
Applicable to your tasks	□	□	□	□	□

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? □ Yes □ No

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.

Name _____ Address _____

Company or Organization _____

Phone No. _____

Readers' Comments — We'd Like to Hear from You
SC33-6716-00

Cut or Fold
Along Line

Fold and Tape | **Please do not staple** | Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Attn: Dept ECJ - BP/003D
6300 Diagonal Highway
Boulder, CO 80301-9151

Fold and Tape | **Please do not staple** | Fold and Tape

Cut or Fold
Along Line

File Number: S370/S390-40
Program Number: 5690-VSE

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC33-6716-00

Spine information:

IBM VSE/Enterprise Systems Architecture System Macros Reference *Version 2 Release 4*