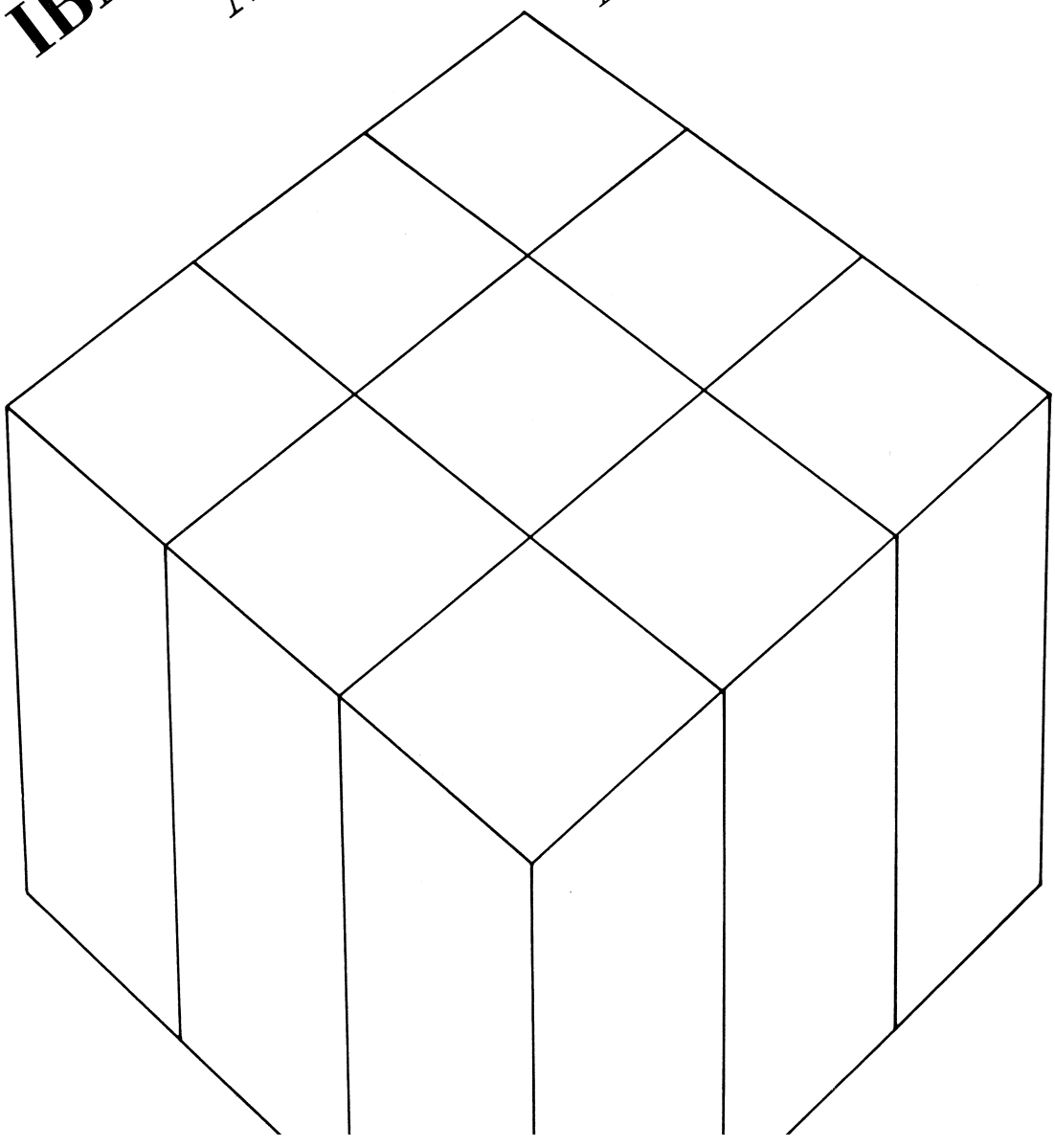
# IBM Virtual Storage Extended Advanced Functions

## Application Programming: Macro Reference



# IBM Virtual Storage Extended/ Advanced Functions

# Application Programming Macro Reference

**Version 2 Release 1**

**Program Number 5666-301**

**Order Number SC33-6197-02**
**File No. S370/4300-30**

**Third Edition (January 1987)**

This edition is a major revision of SC33-6197-1. Changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

This edition applies to Version 2, Release 1 of IBM Virtual Storage Extended/Advanced Functions, Program Number 5666-301, and to all subsequent releases until otherwise indicated in new editions or Technical Newsletters. Changes or additions to the text and illustrations are indicated by a vertical bar to the left of the change. Changes are made periodically to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest IBM System/370, 30XX and 4300 Processors Bibliography, GC20-0001, for the editions that are applicable and current.

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 licensed program in this document is not intended to state or imply that only IBM's licensed program may be used. Any functionally equivalent program may be used instead.

Publications are not stocked at the addresses given below; requests for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for readers' comments is provided at the back of this publication. If the form has been removed, comments may be addressed either to:

IBM Corporation
Dept. 6R1BP
180 Kost Road
Mechanicsburg, PA 17055, USA

or to:

IBM Deutschland GmbH
Dept. 3164
Schoenaicher Strasse 220
D-7030 Boeblingen, Federal Republic of Germany

IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

© Copyright International Business Machines Corporation 1984, 1985, 1987

# PREFACE

The licensed program IBM Virtual Storage Extended/Advanced Functions (VSE/Advanced Functions for short), includes macros that help you in writing application programs (or routines of such programs) in assembler language. This publication documents reference information about these macros, of which there are two categories: data-management and system-control macros. The publication lists the macros in alphabetic order of their names. It includes appendixes as follows:

Appendix A: Printer and stacker-select control characters.
Appendix B: ASCII codes.
Appendix C: Label formats.

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/Advanced Functions Application Programming: Macro User's Guide, SC33-6196.

OS/VS - DOS/VSE - VM/370 Assembler Language, GC33-4010.

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

Using the VSE/VSAM Space Management for SAM Feature, SC24-5192.

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

VSE/Advanced Functions System Management Guide, SC33-6191.

VSE/Advanced Functions System Control Statements, SC33-6198.

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

VSE/Advanced Functions Service Aids, SC33-6195.

For a complete overview of the publications documenting VSE/Advanced Functions, refer to the IBM publication:

VSE/Advanced Functions Planning and Installation, SC33-6193.

## CONTENTS


**Macro Notation** . . . . . 1
Macro Fields . . . . . 1
Notational Conventions . . . . . 2

**Macro Descriptions** . . . . . 5
ASPL (Assign Parameter List) Macro . . . . . 5
ASSIGN (Assign I/O Device) Macro . . . . . 5
ATTACH (Attach a Task) Macro . . . . . 6
CALL (Call a Program) Macro . . . . . 9
CANCEL (Cancel Task) Macro . . . . . 10
CCB (Command Control Block Definition) Macro . . . . . 11
CDLOAD (Control-Directory Load) Macro . . . . . 19
CDMOD (Card I/O Module Definition) Macro . . . . . 21
CHAP (Change Priority) Macro . . . . . 25
CHECK (Check I/O Completion) Macro . . . . . 26
CHKPT (Checkpoint Request) Macro . . . . . 32
CLOSE and CLOSER (Close a File) Macro . . . . . 34
CNTRL (Control Device) Macro . . . . . 35
COMRG (Communication Region Access) Macro . . . . . 41
CPCLOSE (Control Program File Close) Macro . . . . . 41
DEQ (Dequeue Resource) Macro . . . . . 42
DETACH (Detach Task) Macro . . . . . 43
DFR (Define Font Record) Macro . . . . . 44
DIMOD (Device-Independent I/O Module Definition) Macro . . 46
DISEN (Disengage Document Reader) Macro . . . . . 48
DLINT (Define Line Type) Macro . . . . . 49
DRMOD (Document Read Module Definition) Macro . . . . . 52
DSPLY (Display Document Field) Macro . . . . . 53
DTFCD (Define the File for Card I/O) Macro . . . . . 54
DTFCN (Define the File for Console I/O) Macro . . . . . 63
DTFDA (Define the File for Direct Access) Macro . . . . . 66
DTFDI (Define the File for Device Independence) Macro . . 73
DTFDR (Define the File for Document Reader) Macro . . . . 78
DTFDU (Define the File for Diskette Unit I/O) Macro . . . 82
DTFIS (Define the File for Indexed Sequential Access) Macro 88
DTFMR (Define the File for Magnetic Reader Input) Macro . 100
DTFMT (Define the File for Magnetic Tape I/O) Macro . . . 103
DTFOR (Define the File for Optical Reader Input) Macro . . 111
DTFPH (Define the File for Physical I/O) Macro . . . . . 119
DTFPR (Define the File for Printer) Macro . . . . . 124
DTFSD (Define the File for Sequential Disk I/O) Macro . . 132
DTL (Define the Lock) Macro . . . . . 140
DUMODFx (Diskette Unit I/O Module Definition) Macro . . . 142
DUMP (Dump Request) Macro . . . . . 144
ENDFL (End File Load Mode) Macro . . . . . 145
ENQ (Enqueue a Task) Macro . . . . . 145
EOJ (End of Job) Macro . . . . . 146

ERET (Error-Handling Return) Macro . . . . . 147
ESETL (End Set Limit) Macro . . . . . 147
EXCP (Execute Channel Program) Macro . . . . . 148
EXIT (Return from Exit Routine) Macro . . . . . 149
EXTRACT (Extract Control Information) Macro . . . . . 150
FCEPGOUT (Force Page Out) Macro . . . . . 152
FEOV (Force End of Volume) Macro . . . . . 154
FEOVD (Force End of Volume for Disk) Macro . . . . . 155
FETCH (Fetch a Phase) Macro . . . . . 155
FREE (Free Disk Area) Macro . . . . . 158
FREEVIS (Free Virtual Storage) Macro . . . . . 159
GENDTL (Generate the DTL Block) Macro . . . . . 161
GENIORB (Generate an IORB) Macro . . . . . 163
GENL (Generate Directory List) Macro . . . . . 165
GET (Get a Record) Macro . . . . . 166
GETIME (Get the Time) Macro . . . . . 167
GETVIS (Get Virtual Storage) Macro . . . . . 169
IJBPUB (IJBPUB DSECT) Macro . . . . . 171
IORB (I/O Request Block Definition) Macro . . . . . 172
ISMOD (Indexed Sequential I/O Module Definition) Macro . . 175
JDUMP (Job Dump Request) Macro . . . . . 179
JOBCOM (Job Communication) Macro . . . . . 180
LBRET (Label-Routine Return) Macro . . . . . 181
LFCB (Load Forms Control Buffer) Macro . . . . . 183
LITE (Pocket-Light Control) Macro . . . . . 185
LOAD (Load a Phase) Macro . . . . . 186
LOCK (Lock a Resource) Macro . . . . . 190
MAPBDY (Map Boundary Information) Macro . . . . . 194
MAPBDYVR (Map Boundary Information) Macro . . . . . 195
MAPSSID (Map for SUBSID) Macro . . . . . 196
MODDTL (Modify DTL Block) Macro . . . . . 198
MRMOD (MICR Input Module Definition) Macro . . . . . 200
MVCOM (Move to Communication Region) Macro . . . . . 201
NOTE (Note-Address) Macro . . . . . 202
OPEN and OPENR (Open a File) Macro . . . . . 203
ORMOD (Optical Reader Input Module Definition) Macro . . . 204
PAGEIN (Page-In Request) Macro . . . . . 206
PDUMP (Partial-Dump Request) Macro . . . . . 208
PFIX (Page-Fix Request) Macro . . . . . 209
PFREE (Page-Free Request) Macro . . . . . 211
POINTR (Point to Noted Record) Macro . . . . . 212
POINTS (Point to Start) Macro . . . . . 213
POINTW (Point Behind Noted Record) Macro . . . . . 213
POST (Post Event) Macro . . . . . 214
PRMOD (Printer Output Module Definition) Macro . . . . . 215
PRTOV (Printer Overflow Control) Macro . . . . . 220
PUT (Put Record) Macro . . . . . 221
PUTR (PUT with Reply) Macro . . . . . 223
RCB (Resource Control Block Definition) Macro . . . . . 224
RDLNE (Read a Line) Macro . . . . . 225
READ (Read a Record) Macro . . . . . 226
REALAD (Real Address Return) Macro . . . . . 227
RELEASE (Release Logical Unit) Macro . . . . . 228

RELPAG (Release Page) Macro . . . . . 229
RELSE (Release a Block) Macro . . . . . 231
RESCN (Re-Scan) Macro . . . . . 231
RETURN (Return after Call) Macro . . . . . 232
RUNMODE (Run-Mode Indication) Macro . . . . . 233
SAVE (Save Register) Macro . . . . . 233
SECTVAL (Sector-Value Calculation) Macro . . . . . 234
SEOV (System End-of-Volume) Macro . . . . . 235
SETDEV (Set Device) Macro . . . . . 236
SETFL (Set File Load Mode) Macro . . . . . 236
SETIME (Set Interval Timer) Macro . . . . . 237
SETL (Set Limits) Macro . . . . . 238
SETPFA (Set Link to Page-Fault Appendage) Macro . . . . . 240
SETT (Set Task Timer) Macro . . . . . 244
STXIT (Set Exit) Macro . . . . . 245
SUBSID (Subsystem Information Display) Macro . . . . . 251
TECB (Timer Event Control Block) Macro . . . . . 252
TESTT (Test Task Timer) Macro . . . . . 252
TPIN (Telecommunication Priority In) Macro . . . . . 253
TPOUT (Telecommunication Priority Out) Macro . . . . . 253
TRUNC (Truncate Block) Macro . . . . . 254
TTIMER (Test Interval Timer) Macro . . . . . 254
UNLOCK (Unlock Resource) Macro . . . . . 255
VIRTAD (Virtual Address Return) Macro . . . . . 256
WAIT (Wait for Event) Macro . . . . . 256
WAITF (Wait for Completion of I/O) Macro . . . . . 258
WAITM (Wait for Multiple Events) Macro . . . . . 258
WRITE (Write a Record) Macro . . . . . 259
XECBTAB (Cross-Partition Event Control Block Table) Macro 261
XPOST (Cross-Partition Post) Macro . . . . . 264
XWAIT (Cross-Partition Wait) Macro . . . . . 266

**Appendix A. Control Character Codes . . . . . 269**

**Appendix B. American National Standard Code for Information Interchange . . . . . 275**

**Appendix C. Standard and Non-Standard Labels . . . . . 279**
Processing of User Labels . . . . . 279
Formats of Volume and File Labels . . . . . 282

**Index . . . . . 299**

## FIGURES


1. Subtaks Save Area (120 Bytes) . . . . . 8
2. Layout and Contents of the Command Control Block (CCB) 12
3. MICR Document Buffer Format . . . . . 27
4. Character Set Option List . . . . . 45
5. ASOCFLE Operand Usage with Print Associated Files . . . 55
6. Label Extent Information Field . . . . . 72
7. COREXIT Routine Functions . . . . . 79
8. DTFDU Error Options . . . . . 85
9. FilenameC-Status or Condition Code Byte if IOROUT=ADD, RETRVE, or ADDRTR . . . . . 90
10. FilenameC-Status or Condition Code Byte if IOROUT=LOAD 91
11. ERREXT Parameter List . . . . . 92
12. Output Area Requirements for Loading or Adding Records to a File by ISAM . . . . . 95
13. I/O Area Requirements for Random or Sequential Retrieval by ISAM . . . . . 96
14. Work Area Requirements . . . . . 100
15. Operands to Define a Checkpoint File on Disk . . . . . 119
16. ASOCFLE Operand Usage with Print Associated Files . . . 125
17. Maximum and Assumed Lengths for the IOAREA1 . . . . . 126
18. Layout and Contents of the I/O Request Block (IORB) . . 172
19. Bit Configuration of the Pocket-Light Switch Area . . . 186
20. System Action for Control Definitions in DTLs . . . . . 192
21. System Actions by Return Code and FAIL Operand . . . . 193
22. Layout of the MAPBDY-generated DSECT . . . . . 195
23. Layout of the MAPBDYVR-generated DSECT . . . . . 196
24. Layout of MAPSSID-generated DSECT . . . . . 197
25. Field Supplied for SETL Processing by Record ID . . . . 239
26. Internal Page-Fault Queue and Communication with the System . . . . . 243
27. Effect of an AB, IT, OC, PC, or TT Interrupt During STXIT Routine Execution . . . . . 250
28. XECBTAB Feedback Information . . . . . 263
29. Coding Example Showing the Use of XECBTAB with TYPE=CHECK and XWAIT . . . . . 268
30. ASCII Character Set . . . . . 276
31. ASCII to EBCDIC Correspondence . . . . . 277
32. Disk Volume Label (VOL1) . . . . . 282
33. IBM-Standard Disk File Label (Format-1) . . . . . 284
34. IBM-Standard Disk File Continuation Label (Format-3) . 286
35. Disk VTOC Label (Format-4) . . . . . 287
36. User-Standard Disk-File Label (Header and Trailer) . . 290
37. User-Standard Disk-File Label (Five UHLs and Four UTLs are Specified) . . . . . 290
38. User-Standard Disk-File Label (Three UHLs are Specified) 291
39. Diskette Volume Label . . . . . 292
40. Diskette File Label . . . . . 292

41. Tape Volume Label for EBCDIC Code . . . . . 294
42. Tape Volume Label for ASCII Code . . . . . 294
43. IBM-Standard Tape File Label for EBCDIC Code . . . . . 296
44. First IBM-Standard Tape File Label for ASCII Code . . . 296
45. Second IBM-Standard Tape File Label for ASCII Code . . 297

## SUMMARY OF AMENDMENTS

This edition documents the extensions of the CNTRL macro in support of the IBM 3480 and 4248. It includes some minor corrections and editorial improvements.

Sections have been added to describe the coding requirements for:

A routine for the processing of user labels
A routine for page-fault handling overlap.

The previous edition was revised to include changes and additions resulting from new or changed functions as listed below:

- Virtual addressability extension
- New hardware support (IBM 3380)
- GETVIS area subpooling
- 4K paging
- Device-independent full track support
- Larger block size for DTFCD
- End-of-extent exit
- Extended end-of-extent support (DTFPH)
- Adaptation to new librarian
- Tape I/O module in SVA
- Dropped paper tape support
- Performance improvement for PRT1/3800 printers
- CANCEL macro extension
- ANSI security checking

In addition, label formats and field descriptions were added in that edition as Appendix C. This information was taken over from the former IBM manual VSE/Advanced Functions: Disk, Diskette, and Tape Labels, SC24-5213, which has been discontinued.

# MACRO NOTATION

There are 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 to a user of VSE/Advanced Functions.

All macros are written in assembler format statements; they consist of a number of fields as discussed under "Macro Fields" below.

## Macro Fields

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, TESTT 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.

### 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 IBM publication OS/VS-DOS/VSE-VM/370 Assembler Language.

### 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.

## Notational Conventions

The following conventions are used in this book to illustrate the format of macros:

- Uppercase letters and punctuation marks (except as described in these conventions) must be coded exactly as shown.
- Lowercase letters and terms represent information which you must supply. More specifically, an n indicates a number, an r indicates a decimal register number and an x indicates an alphameric character.
- Information contained within brackets [ ] represents an optional value that can be included or omitted, depending on the requirements of the program.

- Options contained within brackets and separated by an OR symbol (|) represent alternatives, one of which may be chosen. For example:

  [,entrypoint|,(0)]

- An underlined option represents the assumed default value in case you omit the operand. For example:

  [A|B|C]

- Options contained within braces { } and separated by an OR symbol (|) represent alternatives, one of which must be chosen. For example:

  {phasename|(1)}

- An ellipsis (a series of three periods) indicates that a variable number of items may be included.

## Register Notation

Certain operands can be specified in either of two ways:

- You may specify the operand directly which 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 the generated code is reentrant and 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.

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 explicit address, that is, an assembler instruction address in base-displacement form. Address can be specified either as a relocatable expression, for example: (S,RELOC), or 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.

## MACRO DESCRIPTIONS

This section of the publication lists the macros in alphabetical order of their names.

### ASPL (Assign Parameter List) Macro

| Name | Operation | Operand |
|---|---|---|
| [name] | ASPL | [DSECT={NO\|YES}] |

The macro generates a five-byte parameter list that is used to pass information to the ASSIGN macro. For the format of the parameter list, see the IBM publication Application Programming: 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, in-line code is generated.

### ASSIGN (Assign I/O Device) Macro

| Name | Operation | Operand |
|---|---|---|
| [name] | ASSIGN | ASPL={name1\|(r1)}<br>,SAVE={name2\|(r2)} |

The macro is used to dynamically assign and unassign tape, disk, and unit-record devices. The system will select a free unit and assign it temporarily to any free programmer logical unit. 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 given in the IBM publication Application Programming: 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.

## ATTACH (Attach a Task) Macro

| Name | Operation | Operand |
|---|---|---|
| [name] | ATTACH | {entrypoint\|(S,entrypoint)\|(r1)}<br>[,SAVE={savearea\|(S,savearea)\|(r2)}]<br>[,ABSAVE={absavearea\|(S,absavearea)\|(r3)}]<br>[,ECB={ecbname\|(S,ecbname)\|(r4)}]<br>[,NAME={name\|(S,name)\|(r5)}]<br>[,MFG={area\|(S,area)\|(r6)}] |

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

The maximum number of subtasks that can be initiated in the system at a time is 208. Up to 31 subtasks can run concurrently within a partition, provided the overall limitation of 208 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, 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. 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. 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.

Upon return from a successful ATTACH, register 0 of the attaching task contains the address of the byte immediately following the save area of the attached task.

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 double words).

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 in this area data as follows (for the format of the area, see Figure 1 on page 8):

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

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.

| 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 |

Figure 1. Subtaks Save Area (120 Bytes)

ABSAVE={absavearea|(S,absavearea)|(r3)}
Specify this operand only if the subtask is to use the attaching task's abnormal termination routine (see STXIT macro), 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 a 72-byte (doubleword-aligned) STXIT save area for the subtask. When an abnormal termination occurs, the supervisor saves the old PSW and general registers 0 through 15 in this area before the exit is taken.

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, RETURN, DUMP, JDUMP, or EOJ.

NAME={name|(S,name)|(r5)}
You can specify the subtask name here 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.

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 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.

**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.

## CALL (Call a Program) Macro

| Name | Operation | Operand |
|---|---|---|
| [name] | CALL | {entrypoint\|(15)}<br>[,(parameterlist)] |

The macro passes control from one program to a specified entry point in another program.

entrypoint|(15)
Specifies the entry point to which control is passed. If the symbolic name of an entry point is specified, an instruction

```
L 15,=V(entrypoint)
```

is generated as part of the macro expansion. The linkage editor makes the called program part of the calling program phase. The symbolic name must be either the name of a control section (CSECT) or an assembler language ENTRY statement operand in the called program. Control is given to the called program at this address. The called program resides in storage throughout execution of the calling program. This wastes storage if the called program is not needed throughout execution of the calling program.

If register 15 is specified, the address of the entry point must have been loaded into that register. Control is given to the called program at the address in register 15. Specifying register 15 preceded by a LOAD macro is most useful when the

same program is called many times during execution of the calling program, but is not needed in storage throughout execution of the calling program.

parameterlist
: Specifies one or more addresses (relocatable or absolute expressions) to be passed to the called program. Terms in the address must not be indexed. The addresses must be written in a sublist, with each address separated from the next by a comma. As part of the macro expansion, a parameter list is generated. It consists of a fullword for each address. Each fullword is aligned on a fullword boundary and contains the address to be passed in its three low-order bytes. The high-order bit in the last fullword is set to 1. When the called program is entered, register 1 contains the address of the parameter list.

## CANCEL (Cancel Task) Macro

| Name | Operation | Operand |
|---|---|---|
| [name] | CANCEL | [ALL] |

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. Upon task termination, system messages (using the subtask name) are issued to identify each subtask terminated.

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

| Name | Operation | Operand |
|---|---|---|
| blockname | CCB | SYSnnn,command-list-name [,X'nnnn'][,senseaddress] |

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 12) 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.

blockname
: 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 14. 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 16) 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.

## 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.

| 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 on page 13). |
| 6 | 2 | Type code (see Note 4 on page 13). |
| 8 | 1 | Reserved. |
| 9 | 3 | Address of CCW (see Note 5 on page 13). |
| 0C | 1 | Reserved. |
| 0D | 3 | CCW address in CSW (see Note 6 on page 14). |
| 10 | 10 | Optional sense CCW (see Note 7 on page 14). |

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

**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. The field is set to zero for negative values.

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 14.

   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, each 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 |

If bit 5 of CCB byte 2 is set to 1 and device end results as a separate interrupt, device end will be posted.

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.

Mask for TM instruction: X'80'.

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 19) - a UCB parity check (line complete).

Mask for TM instruction: X'40'.

Bit 2 - Unrecoverable 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.

Mask for TM instruction: X'20'.

Bit 3 - Accept unrecoverable I/O error - Bit 2 is set to 1 (see also Note 2 on page 19):

If 0 Operator option - depends on the error.
If 1 Return to the user after physical IOCS attempted to correct the error. The cause of the I/O error is set into bytes 4 and 5 of the CCB.

Mask for TM instruction: X'10'.

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

Bit 4 - Return

- Disk data check
- Data check on an IBM 3540
- Not ready for an IBM 5424 or 5425.
- 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.

Mask for TM instruction: X'08'.

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

Bit 5 - Post at device end:

Specify this bit to be set on for an IBM 2560, 5424, or 5425 (see also Note 2 on page 19).

If 0 Device-end condition is not posted. The traffic bit (bit 0) is set at channel end.
If 1 Device-end condition is posted: bit 0 of this byte and bits 2 and 6 of byte 3 are set to 1 at device end. In addition, bit 5 of byte 4 (device end) is set to 1.

Mask for TM instruction: X'04'.

ON value for the third operand of the CCB macro: X'0400'.

Bit 6 - Return on:

- DASD read or read-verify data check.
- PRT1 printer passback requested (see also Note 1 on page 19).
- Tape read data check.
- Punch equipment check on an IBM 2520 or 2540.
- Data check on an IBM 2560
- Equipment check (read, punch, print data, and print clutch) on an IBM 2560, 5424, or 5425.
- 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 other than IBM 2560, 5424, and 5425.
- Retry or cancel for DASD and for IBM 2560, 5424, and 5425.

If 1 For an error on a PRT1 printer (see also Note 1 on page 19), 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.

Mask for TM instruction: X'02'.

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

Bit 7 - User error routine (see also Note 2 on page 19):

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. I/O program and protection checks always cause program termination. 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.

Mask for TM instruction: X'01'.

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 MICR device - SCU not operational.
- For an IBM 1287 or 1288 - data check.
- For a PRT1 printer (see also Note 1 on page 19) - 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.

Mask for TM instruction: X'80'.

Bit 1 - See "If 1" below:

If 1 Indicates one of the following:
- DASD track overrun.
- For an IBM 1287 in journal-tape mode - keyboard correction.
- For a MICR device - intervention required.
- For a PRT1 printer (see also Note 1 on page 19) - a print quality error (equipment check).

Mask for TM instruction: X'40'.

Bit 2 - See "If 1" below:

If 1 Indicates one of the following:
- End of DASD cylinder.
- For an IBM 1287 or 1288 in document mode - hopper empty.
- For an IBM 1255, 1270, 1275, or 1419 - disengaged; feeding of documents has stopped.
- For an IBM 1275 or 1419D - I/O error in the external-interrupt routine (channel-data or busout check.
- For a PRT1 printer (see also Note 1 on page 19) - a line-position error. A line-position error can occur as a result of an equipment, data, or FCB-parity check.

Mask for TM instruction: X'20'.

Bit 3 - See "If 1" below:

If 1 Indicates one of the following:
- Tape read data check (see "Note" below).
- For an IBM 2520, 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 1287 or 1288 - equipment check.
- For an IBM 2560, 3203, 5424 or 5425 - 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 19) - a data/print check.
- For an IBM 3540 - data check.

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

Mask for TM instruction: X'10'.

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 MICR device - SCU not operational.
- For an IBM 1287 or 1288 - document jam or torn tape.
- For a PRT1 printer (see also Note 1 on page 19) - UCB parity check (command retry).
- For an IBM 5424 or 5425 - device not ready.

Mask for TM instruction: X'08'.

Bit 5 - No record found:

Applies to a retry on DASD (see also Note 2 on page 19).

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.

Mask for TM instruction: X'04'.

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

Bit 6 - See "If 1" below and also Note 2 on page 19):

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.
- Late stacker select for an IBM 1287 in document mode.
- End of page for an IBM 1288.

Mask for TM instruction: X'02'.

Bit 7 - Command-chain retry:

Specify the bit to be set to 1 if you use command chaining (see also Note 2 on page 19).

If 0 Retry begins at the first CCW (or channel program).
If 1 Retry begins at the last CCW that was executed. However:
- For a tape error retry, the whole CCW chain is retried.
- 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.

Mask for TM instruction: X'01'.

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.

## CDLOAD (Control-Directory Load) Macro

| Name | Operation | Operand |
|---|---|---|
| [name] | CDLOAD | {phasename\|(1)}<br>[,PAGE={NO\|YES}]<br>[,RETPNF={NO\|YES}] |

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 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.

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.

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), registers contain values as follows:

Register 0: the load address.
Register 1: the entry point.
Register 14: the length of the phase.

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.

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.

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 OK. |
| 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. |
| 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

| Name | Operation | Operand |
|---|---|---|
| [name] | CDMOD | [CONTROL=YES]<br>[,CRDERR=RETRY]<br>[,CTLCHR={ASA\|YES}]<br>[,DEVICE=nnnn]<br>[,FUNC={R\|P\|I\|RP\|RW\|RPW\|PW}]<br>[,IOAREA2=YES]<br>[,RDONLY=YES]<br>[,RECFORM={FIXUNB\|VARUNB\|UNDEF}]<br>[,SEPASMB=YES]<br>[,TYPEFLE={INPUT\|OUTPUT\|CMBND}]<br>[,WORKA=YES] |

The CDMOD macro defines a logic module for a card reader/punch file. 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 2560, 3525, or 5424/5425; 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). 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 | 2560P | 3881 |
| 1442 | 2560S | 5425P |
| 2501 | 3505 | 5425S |
| 2520 | 3525 | |

Include this operand to specify the I/O device used by the module. The characters 'P' and 'S' with '2560' and '5425' specify primary or secondary input hoppers; however, regardless of which is specified, the generated module handles DTFs specifying either hopper. Specify 5425P or 5425S for the primary or secondary hopper of an IBM 5424.

Any DTF to be used with this module must have the same operand (except as just noted concerning the 'P' and 'S' specification for the IBM 2560 or 5425).

FUNC={R|P|I|RP|RW|RPW|PW}
This operand specifies the type of file to be processed by the IBM 2560, 3525, 5424, or 5425. 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

```
a = F   RECFORM=FIXUNB (always for INPUT, CMBND, or FUNC=I files)
  = V   RECFORM=VARUNB
  = U   RECFORM=UNDEF

b = A   CTLCHR=ASA (not specified if CMBND)
  = Y   CTLCHR=YES
  = C   CONTROL=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 = Z   WORKA and IOAREA2 not specified
  = W   WORKA=YES
  = I   IOAREA2=YES
  = B   WORKA and IOAREA2
  = Z   WORKA=YES not specified (CMBND file only)
```

```
e = 0   DEVICE=2540
  = 1   DEVICE=1442, 2596
  = 2   DEVICE=2520
  = 3   DEVICE=2501
  = 4   DEVICE=2540 and CRDERR
  = 5   DEVICE=2520 and CRDERR
  = 6   DEVICE=3505
  = 7   DEVICE=3525 and FUNC=R/P or omitted
  = 8   DEVICE=2560 and FUNC=R/P or omitted
  = 9   DEVICE=5425 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
  = F   DEVICE=2560 and FUNC=RP
  = G   DEVICE=2560 and FUNC=RW
  = H   DEVICE=2560 and FUNC=PW
  = I   DEVICE=2560 and FUNC=I
  = J   DEVICE=2560 and FUNC=RPW
  = K   DEVICE=5425 and FUNC=RP
  = L   DEVICE=5425 and FUNC=RW
  = M   DEVICE=5425 and FUNC=PW
  = N   DEVICE=5425 and FUNC=I
  = O   DEVICE=5425 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.

| | | |

## CHECK (Check I/O Completion) Macro

| Name | Operation | Operand |
|---|---|---|
| [name] | CHECK | {filename\|(1)}<br>[,control-address\|,(0)] |

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 27) 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.

| Byte | Bit | Comment |
|---|---|---|
| 0 | | Buffer Status Indicator |
| | 0 | The document is ready for processing (you need not test this bit). |
| | 1 | Unrecoverable stacker select error, but all document data is present. You may continue to issue GETs and READs. |
| | 2 | Unrecoverable 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. |

Figure 3 (Part 1 of 5). MICR Document Buffer Format

| Byte | Bit | Comment |
|---|---|---|
| 0 | 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 has issued a GET or READ and the file is closed. Bit 5 also on. |
| | 7 | Reserved. |
| 1 | | Buffer Status Indicator |
| | 0 | Your stacker selection routine turns on this bit to indicate that batch numbering update (IBM 1419 only) is to be performed together with stacker selection for this document. The document is imprinted with the updated batch number, except if a late stacker selection occurs (bit 2 of byte 3 is on). If bit 6 or bit 7 of byte 2 is on, this bit is ignored by the external interrupt routine. With the IBM 1419 (dual address only), batch numbering update cannot be performed by stacker selection of auto-selected documents. |
| | 1-7 | Reserved. |
| 2 | | Buffer Status Indicator |
| | 0 | For an IBM 1419 or 1275 (dual address) only. An auto-select condition occurred at the completion of a READ but before a stacker select command. The document is auto-selected into the reject pocket. |
| | 1-3 | Reserved. |
| | | 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. When the IBM 1419 or 1275 is enabled for fixed-length data fields, bit 4 is set. |

Figure 3 (Part 2 of 5). MICR Document Buffer Format

| Byte | Bit | Comment |
|---|---|---|
| 2 | 6-7 | The meanings of these bits depend on the device type, the model, and the engineering-change level of the MICR device. If either bit is on, the involved document(s) is (are) auto-selected into the reject pocket.<br><br>1. For an IBM 1412 or 1270, if:<br><br>Bit 6 is on – A late read condition occurred.<br>Bit 7 is on – A document spacing error occurred.<br><br>2. For an IBM 1270 only:<br>Both the current and the previous document are auto-selected into the reject pocket if Bit 7 is on. This previous document reject cannot be detected by IOCS, and byte 5 of its document buffer does not reflect that the reject pocket was selected.<br><br>3. For an IBM 1275 and 1419 (single address) without engineering change 125358:<br><br>If Bit 6 is on – A late read condition or a document spacing error occurred.<br><br>If Bit 7 is on – A document spacing error occurred for the current document.<br><br>4. For an IBM 1255, 1275, or 1419 (single or dual address) with engineering change 125358:<br>If Bit 6 is on, an auto-select condition occurred while a document was being read. The bit is set after the READ command and before the stacker select routine receives control. Bit 7 is always zero. |
| 3 | | Buffer Status Indicator – The byte contains MICR sense information.<br><br>For the IBM 1270, Bits 3 through 7 are set to zero when the fields are read without error.<br><br>For the IBM 1255, 1275, and 1419, Bits 3 through 7 are set on when each respective field, including bracket symbols, is read without error. This applies to Bits 0, 1, and 3 through 7 on the 1419 Model 32. |

Figure 3 (Part 3 of 5). MICR Document Buffer Format

| Byte | Bit | Comment |
|---|---|---|
| 3 | | For an IBM 1255, 1275, and 1419, unread fields contain zero bits. Errors are indicated when an overrun or data check condition occurs while the data field is being read. |
| | 0 | Field 6 is valid. Applies only to an IBM 1419 Model 32. |
| | 1 | Field 7 is valid. Applies only to an IBM 1419 Model 32. |
| | 2 | A late stacker selection (unit check, late stacker select on the stacker select command). The document is auto-selected into the reject pocket. |
| | 3 | The amount field (or field 1) is valid. Applies only to an IBM 1419 Model 32. |
| | 4 | The process-control field (or field 2) is valid. Applies only to an IBM 1419 Model 32. |
| | 5 | The account-number field (or field 3) is valid. Applies only to an IBM 1419 Model 32. |
| | 6 | The transit field (or field 4) is valid. Applies only to an IBM 1419 Model 32. |
| | 7 | The serial number field (or field 5) is valid. Applies only to an IBM 1419 Model 32. |
| 4 | | Buffer Status Indicator - Pocket code<br><br>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.<br><br>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: |

Figure 3 (Part 4 of 5). MICR Document Buffer Format

| Byte | Bit | Comment |
|---|---|---|
| 4 | | Pocket A: X'AF' * Pocket 5: X'5F'<br>Pocket B: X'BF' ** Pocket 6: X'6F' ***<br>Pocket 0: X'0F' Pocket 7: X'7F'<br>Pocket 1: X'1F' Pocket 8: X'8F'<br>Pocket 2: X'2F' Pocket 9: X'9F'<br>Pocket 3: X'3F' Reject Pocket: X'CF'<br>Pocket 4: X'4F'<br><br>* Only for IBM 1270 Models 2 and 4, 1275, and 1419.<br>** Only for IBM 1275 and 1419<br>*** Except for IBM 1270 Models 1 and 3 |
| 5 | | Buffer Status Indicator – Pocket-selected code<br><br>Indicates the actual pocket selected for the document. The contents of the byte are normally the same as that in byte 4.<br><br>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:<br><br>• Start I/O for stacker selection is unsuccessful (bit 1 of byte 0).<br>• An I/O error occurs (invalid pocket code, for example) on the secondary control unit of a (dual address) IBM 1419 when selecting this document. |
| 6-end | | Additional User Work Area<br><br>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.<br><br>Document Data Area<br><br>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. |

Figure 3 (Part 5 of 5). MICR Document Buffer Format

## CHKPT (Checkpoint Request) Macro

| Name | Operation | Operand |
|---|---|---|
| [name] | CHKPT | SYSnnn,{restart-address\|(r1)}<br>[,end-address\|,(r2)]<br>[,tpointer\|,(r3)]<br>[,dpointer\|,(r4)]<br>[,filename\|,(r5)] |

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 IBM publication System Management Guide.

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.

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.

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.

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

| Name | Operation | Operand |
|---|---|---|
| [name] | CLOSE | {filename1\|(r1)}<br>[,filename2\|,(r2)]... |

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.

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.

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.

filename|(rn)
: Code the symbolic name of the file (DTF filename) to be closed. 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 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

| Name | Operation | Operand |
|---|---|---|
| [name] | CNTRL | {filename\|(1)},code[,[n1][,n2]] |

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.

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. Applies also to the 3350 operating in 3330 compatibility mode. |

### 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. |
| | | | **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. |
| 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. |

### 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, 1443, 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. |

| Code | n1 | n2 | Requested Operation |
|---|---|---|---|
| | 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.<br><br>Horizontal copy printing is turned off automatically when one of the following occurs:<br><br>• Your program issues a CLOSE (or CLOSER) macro for the printer file.<br>• 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 1442 and 2520. |
| E | | | Eject to stacker 1. Applies to IBM 1442 only. |
| 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. |
| SS | 1 to 5 | | Select stacker, one of 1 through 5. Applies to IBM 2560 only. |
| SS | 1 to 4 | | Select stacker, one of 1 through 4. Applies only to IBM 5424 and 5425. |

### IBM 3881 Optical Mark Reader

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

### IBM 1287 Optical Reader

| Code | n1 | n2 | Requested Operation |
|---|---|---|---|
| MARK | | | Mark error line in tape mode. |
| READKB | | | Read from the 1287 keyboard in tape mode. |
| EJC | | | Eject document. |
| SSD | 1 to 4 | | Select stacker A, B, reject, or alternate stacking mode. |
| ESD | 1 to 4 | | Eject document and select stacker. |
| INC | 1 to 4 | | Increment document at read station. |

### IBM 1288 Optical Page Reader

| Code | n1 | n2 | Requested Operation |
|---|---|---|---|
| ESD | 1 | | Select stacker A. |
| | 3 | | Select reject (R) stacker. |

### IBM 3886 Optical Character Reader

| Code | n1 | n2 | Requested Operation |
|---|---|---|---|
| INC | | | Increment document at read station. |

| Code | n1 | n2 | Requested Operation |
|---|---|---|---|
| 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

| Name | Operation | Operand |
|---|---|---|
| [name] | COMRG | [REG=(r1)] |

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

| Name | Operation | Operand |
|---|---|---|
| [name] | CPCLOSE | [arglist\|(r1)] |

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

arglist|(r1)
This operand specifies a 16-byte argument 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 argument 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 argument list and this address is loaded into register 1. If no operand is specified, register 1 is assumed

to contain the address of the argument 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

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

## DEQ (Dequeue Resource) Macro

| Name | Operation | Operand |
|---|---|---|
| [name] | DEQ | {rcbname\|(0)} |

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. In such cases, the highest priority task either obtains or maintains control. 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.

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

## DETACH (Detach Task) Macro

| Name | Operation | Operand |
|---|---|---|
| [name] | DETACH | [SAVE={savearea\|(1)}] |

The macro terminates execution of a task. A subtask is normally terminated by issuing a DETACH macro in the main task 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, and the highest priority task obtains control.

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.

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.

**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

| Name | Operation | Operand |
|---|---|---|
| [name] | DFR | FONT=code<br>[,BCH=n]<br>[,BCHSER=n]<br>[,CHRSET=n]<br>[,EDCHAR=(x,...)]<br>[,ERASE={NO\|YES}]<br>[,NATNHP={NO\|YES}]<br>[,REJECT=x] |

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 indi-

cates 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.

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

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

Figure 4. Character Set Option List

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 IBM publication OS/VS-DOS/VSE-VM/370 Assembler Language.

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

| Name | Operation | Operand |
|---|---|---|
| [name] | DIMOD | [IOAREA2=YES]<br>[,RDONLY=YES]<br>[,SEPASMB=YES]<br>[,TRC=YES]<br>[,TYPEFLE={OUTPUT\|INPUT}] |

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

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

| Name | Operation | Operand |
|---|---|---|
| [name] | DISEN | {filename\|(1)} |

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 27).

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

| Name | Operation | Operand |
|---|---|---|
| [name] | DLINT | LFR=n,LINBEG=n<br>[,IMAGE={NO\|YES}]<br>[,NOSCAN=(n,...)]<br>[,FLDn=(field information)]<br>[,EDITn=(code\|EDCHAR\|code,EDCHAR)]<br>[,FREND={NO\|YES}] |

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. Its format is:

```
FLDn=({m|x},[length][,NCRIT|,fontcode|,NCRIT,fontcode])
```

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 IBM publication OS/VS-DOS/VSE-VM/370 Assembler Language.

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 44.

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.

## DRMOD (Document Read Module Definition) Macro

| Name | Operation | Operand |
|---|---|---|
| [name] | DRMOD | [,DEVICE=3886]<br>[,RDONLY=YES]<br>[,SEPASMB=YES]<br>[,SETDEV=YES] |

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

| Name | Operation | Operand |
|---|---|---|
| [name] | DSPLY | {filename\|(1)},(r2),(r3) |

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 on-line correction mode) or by the device (if processing in the off-line 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.

## DTFCD (Define the File for Card I/O) Macro

| Name | Operation | Operand |
|---|---|---|
| [name] | DTFCD | DEVADDR=SYSxxx |
| | | ,IOAREA1=name |
| | | [,ASOCFLE=filename] |
| | | [,BLKSIZE=n] |
| | | [,CONTROL=YES] |
| | | [,CRDERR=RETRY] |
| | | [,CTLCHR={ASA\|YES}] |
| | | [,DEVICE=nnnn] |
| | | [,EOFADDR=name] |
| | | [,ERROPT={IGNORE\|SKIP\|name}] |
| | | [,FUNC=xxx] |
| | | [,IOAREA2=name] |
| | | [,IOREG=(r)] |
| | | [,MODE=xx] |
| | | [,MODNAME=name] |
| | | [,OUBLKSZ=n] |
| | | [,RDONLY=YES] |
| | | [,RECFORM={FIXUNB\|VARUNB\|UNDEF}] |
| | | [,RECSIZE=(r)] |
| | | [,SEPASMB=YES] |
| | | [,SSELECT=n] |
| | | [,TYPEFLE={INPUT\|OUTPUT\|CMBND}] |
| | | [,WORKA=YES] |

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 2560, 3525, 5424, or 5425. For a discussion of associated files see the IBM publication Application Programming: Macro User's Guide. ASOCFLE specifies the file name of associated read, punch, or print files as shown in Figure 5. 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.

| Your FUNC= Operand | Filename Specification in ASOCFLE= Operand of | | |
|---|---|---|---|
| | Read DTFCD | Punch DTFCD | Print DTFPR |
| FUNC=PW | | Filename of print DTFPR | Filename of punch DTFCD |
| FUNC=RP | Filename of punch DTFCD | Filename of read DTFCD | |
| FUNC=RPW | Filename of punch DTFCD | Filename of print DTFPR | Filename of read DTFCD |
| FUNC=RW | Filename of print DTFCD | print DTFPR | Filename of read DTFCD |

Examples:

1. If FUNC=PW is specified, then specify:

   a. The file name of the print DTFPR in the ASOCFLE= operand of the punch DTFCD.
   b. The file name of the punch DTFCD in the ASOCFLE= operand of the print DTFPR.

2. If FUNC=RPW is specified, then specify:

   a. The file name of the punch DTFCD in the ASOCFLE= operand of the read DTFCD.
   b. The file name of the print DTFPR in the ASOCFLE= operand of the punch DTFCD.
   c. The file name of the read DTFCD in the ASOCFLE= operand of the print DTFPR.

Figure 5. 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 2560, 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 2560 or 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 2560, 3525, 5424, or 5425; 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 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 1442, 2520, or 2540 (TYPEFLE=CMBND)
- For an associated file on an IBM 2560, 3525, 5424, or 5425 (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:

| | | |
|---|---|---|
| 2540 | 2560S | 5425P |
| 1442 | 3505 | 5425S |
| 2501 | 3525 | 3881 |
| 2560P | | |

The 'P' and 'S' included with the '2560' and '5425' device codes specify primary or secondary hoppers. Specify 5425P or 5425S for files in the primary or secondary hopper of a 5424. 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). If the system logical units SYSIPT and SYSRDR are assigned to an IBM 5424 or 5425, IOCS requires that the /* card, indicating end-of-file, be followed by a blank card. An error condition results if cards are allowed to run out without a /* trailer card (and without a /& card to indicate end-of-job).

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 on an IBM 2560, 5424, or 5425. 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 17). 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 (in the same card path for an IBM 2560, 5424, or 5425). If the file is an associated file, PUT macros may not be issued for cards in the same device (same card path).

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 2560, 3525, 5424, or 5425.

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 2560, 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 2560: E and C.
- 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 OMR considerations in the IBM publication Application Programming: 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. For double buffering support for the IBM 2501 Card

Reader, the save area must have a length of 76 bytes (to include bytes 0 through 3 of the second CCB).

Each 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. On

an IBM 5424 or 5425, cards from hopper 1 are placed in stacker 1 and cards from hopper 2 are placed in stacker 5 (or 4).

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 2560, 3525, 5424, or 5425.

In this case, SSELECT=n may be specified for the associated output file.

When this operand is used with a device other than an IBM 1442, 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 1442 or 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

| Name | Operation | Operand |
|---|---|---|
| [name] | DTFCN | DEVADDR=SYSxxx<br>,IOAREA1=name<br>[,BLKSIZE=n]<br>[,INPSIZE=n]<br>[,MODNAME=name]<br>[,RECFORM={FIXUNB\|UNDEF}]<br>[,RECSIZE=(r)]<br>[,TYPEFLE={INPUT\|OUTPUT\|CMBND}]<br>[,WORKA=YES] |

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

| Name | Operation | Operand |
|---|---|---|
| [name] | DTFDA | BLKSIZE=n<br>,ERRBYTE=name<br>,IOAREA1=name<br>,SEEKADR=name<br>,TYPEFLE={INPUT\|OUTPUT}<br>[,AFTER=YES]<br>[,CONTROL=YES]<br>[,DEVADDR=SYSnnn]<br>[,DSKXTNT=n]<br>[,ERREXT=YES]<br>[,FEOVD=YES]<br>[,HOLD=YES]<br>[,IDLOC=name]<br>[,KEYARG=name]<br>[,KEYLEN=n]<br>[,LABADDR=name]<br>[,READID=YES]<br>[,READKEY=YES]<br>[,RECFORM=xxxxxx]<br>[,RECSIZE=(r)]<br>[,RELTYPE={DEC\|HEX}]<br>[,SEPASMB=YES]<br>[,SRCHM=YES]<br>[,TRLBL=YES]<br>[,VERIFY=YES]<br>[,WRITEID=YES]<br>[,WRITEKY=YES]<br>[,XTNTXIT=name] |

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 IBM publication Application Programming: 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 the IBM publication Application Programming: Macro User's Guide.

ERREXT=YES
This operand enables unrecoverable 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 the IBM publication Application Programming: 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 279.

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 definition of record formats, see the IBM publication Application Programming: 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.

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 6.

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.).

| 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. |

Figure 6. Label Extent Information Field

## DTFDI (Define the File for Device Independence) Macro

| Name | Operation | Operand |
|---|---|---|
| [name] | DTFDI | DEVADDR=SYSxxx<br>,IOAREA1=name<br>[,CISIZE=n]<br>[,EOFADDR=name]<br>[,ERROPT={IGNORE\|SKIP\|name}]<br>[,IOAREA2=name]<br>[,IOREG=(r)]<br>[,MODNAME=name]<br>[,RDONLY=YES]<br>[,RECSIZE=n]<br>[,SEPASMB=YES]<br>[,TRC=YES]<br>[,WLRERR=name] |

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. The logical device SYSLST must not be assigned to the 2560 or 5424/5425.

EOFADDR=name
: This operand specifies the name of your end-of-file routine. It is required only if SYSIPT or SYSRDR is specified. 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.

If the system logical units SYSIPT and SYSRDR are assigned to a 5424/5425, IOCS requires that the /* card, indicating end-of-file, be followed by a blank card. An error condition results if the records are allowed to run out without a /* card (and without a /& card, if end-of-job). 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; for 2560 or 5424/5425 output files, normal error recovery procedures are followed.

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.

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. You should always specify the maximum of 81 bytes (and an I/O area of 81 bytes) to ensure device independence when reading data. The first byte of the I/O area will always contain the first data byte, regardless of whether the input consisted of 80 data bytes and one control character or of 80 data bytes only. 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 codes for the IBM 2520 cannot be used for a file on the IBM 1442.

- 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.

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

| Name | Operation | Operand |
|---|---|---|
| [name] | DTFDR | COREXIT=name<br>,DEVADDR=SYSxxx<br>,EOFADDR=name<br>,EXITIND=name<br>,FRNAME=name<br>,FRSIZE=n<br>,HEADER=name<br>,IOAREA1=name<br>[,BLKSIZE=nnn]<br>[,DEVICE=3886]<br>[,MODNAME=name]<br>[,RDONLY=YES]<br>[,SEPASMB=YES]<br>[,SETDEV=YES] |

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 7 on page 79 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.

| 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 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). | Routine in your program to read the next document. |
| X'F3' | Rescan the line, using another format record or using image processing and editing the record in your program (see also Note 2). | 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). |

Figure 7. COREXIT Routine Functions

**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 phase name of the format record to be loaded when the file is opened.

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

| Name | Operation | Operand |
|---|---|---|
| [name] | DTFDU | EOFADDR=name<br>,IOAREA1=name<br>,RECSIZE=n<br>[,CMDCHN=n]<br>[,DEVADDR=SYSxxx]<br>[,DEVICE=3540]<br>[,ERREXT=YES]<br>[,ERROPT={IGNORE\|SKIP\|name}]<br>[,FEED={YES\|NO}]<br>[,FILESEC=YES]<br>[,IOAREA2=name]<br>[,IOREG=(r)]<br>[,MODNAME=name]<br>[,RDONLY=YES]<br>[,SEPASMB=YES]<br>[,TYPEFLE={INPUT\|OUTPUT}]<br>[,VERIFY=YES]<br>[,VOLSEQ=YES]<br>[,WORKA=YES]<br>[,WRTPROT=YES] |

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 4-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 8 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 8. 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 disk drive (providing another diskette is still in the hopper). If NO is specified, the diskette 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 multivolume 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

| Name | Operation | Operand |
|---|---|---|
| [name] | DTFIS | DSKXTNT=n<br>,IOROUT=xxxxxx<br>,KEYLEN=n<br>,NRECDS=n<br>,RECFORM={FIXUNB\|FIXBLK}<br>,RECSIZE=n<br>[,CYLOFL=n]<br>[,DEVICE=nnnn]<br>[,ERREXT=YES]<br>[,HINDEX=nnnn]<br>[,HOLD=YES]<br>[,INDAREA=name]<br>[,INDSKIP=YES]<br>[,INDSIZE=n]<br>[,IOAREAL=name]<br>[,IOAREAR=name]<br>[,IOAREAS=name]<br>[,IOAREA2=name]<br>[,IOREG=(r)]<br>[,IOSIZE=n]<br>[,KEYARG=name]<br>[,KEYLOC=n]<br>[,MODNAME=name]<br>[,MSTIND=YES]<br>[,RDONLY=YES]<br>[,SEPASMB=YES]<br>[,TYPEFLE={RANDOM\|SEQNTL\|RANSEQ}]<br>[,VERIFY=YES]<br>[,WORKL=name]<br>[,WORKR=name]<br>[,WORKS=YES] |

The macro defines a disk file for the Indexed Sequential Access Method.

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 unrecoverable 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. Filename is the same 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 9 on page 90 shows the meaning if IOROUT=ADD, RETRVE, or ADDRTR was specified; Figure 10 on page 91 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.

| Bit | Meaning if Set to 1 |
|---|---|
| 0 | Disk error<br>An irrecoverable disk error has occurred (except wrong-length record.) |
| 1 | Wrong-length record<br>A wrong length record has been detected during an I/O operation. |
| 2 | End of file<br>End of file has been encountered during sequential retrieval. |
| 3 | No record found<br>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 | Illegal ID specified<br>The ID specified to the SETL in SEQNTL (RANSEQ) is outside the prime file limits. |
| 5 | Duplicate record<br>The record to be added to the file has a duplicate record key of another record in the file. |
| 6 | Overflow area full<br>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<br>The record being processed in one of the retrieval functions (RANDOM/SEQNTL) is an overflow record. |

Figure 9. FilenameC-Status or Condition Code Byte if IOROUT=ADD, RETRVE, or ADDRTR

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 11 on page 92. After testing filenameC and finding an error, return to IOCS by using the ERET imperative macro; IOCS takes the action indi-

cated by the ERET operand. If HOLD=YES (and ERREXT=YES), ERET must be used to return to IOCS to free any held track.

| Bit | Meaning if Set to 1 |
|---|---|
| 0 | Disk error<br>An irrecoverable disk error has occurred (except wrong-length record.) |
| 2 | Prime area full<br>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<br>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. Extent the upper limit of the cylinder index by using a new EXTENT statement. |
| 4 | Master index full<br>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<br>The record to be added to the file has a duplicate record key of another record in the file. |
| 6 | Sequence check<br>The record being loaded is not in sequential order. |
| 7 | Prime data area overflow<br>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 10. FilenameC-Status or Condition Code Byte if IOROUT=LOAD

| 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:<br>m = Extent sequence number<br>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:<br>Bit 1: Data record<br>Bit 2: Track-index record.<br>Bit 3: Cylinder- or master-index record.<br>Bits 4-5: Reserved.<br>Bit 6: Read operation.<br>Bit 7: Write operation |
| 17 | Command code of failing CCW. |

Figure 11. ERREXT Parameter List

In your program, you should check byte 16, bit 7 of the DTF 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.

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 unrecoverable 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 12 on page 95).

  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 13 on page 96).

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 13 on page 96).

| Function | Output Area Requirements (in 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<br>The greater of: | 8 | + key length + | | | 0 | + | R x B | |
| | 8 | + key length + | | | 10 | + | R | |

B = Blocking factor
R = Record length

Figure 12. 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.

| Function | I/O Area Requirements (in Bytes) | | | |
|---|---|---|---|---|
| | Count | Key | Seq Link | Data |
| Retrieve unblocked records | 0 | + key length (see *) | + 10 | + R |
| Retrieve blocked records – The greater of: | 0 + 0 | + 0 | + 0 | + R x B (see **) |
| | 0 + 0 | + 0 | + 10 | + R |

B = Blocking factor
R = Record length
* Only for sequential retrieval
** Including keys

Figure 13. 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 12 on page 95 and 96).

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 Macro').

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 14.

| Function | Unblocked Records | Blocked Records |
|---|---|---|
| Load - The greater of: | 10<br>K + D | 10<br>D |
| Add - The greater of: | 10<br>K + D | 10<br>K + D |
| Random Retrieve | D | D |
| Sequential Retrieve | K + D | D |
| K = Length of the key<br>D = Length of the data | | |

Figure 14. Work Area Requirements

## DTFMR (Define the File for Magnetic Reader Input) Macro

| Name | Operation | Operand |
|---|---|---|
| [name] | DTFMR | DEVADDR=SYSxxx<br>,IOAREA1=name<br>[,ADDAREA=n]<br>[,ADDRESS=DUAL]<br>[,BUFFERS={25\|n}]<br>[,ERROPT=name]<br>[,EXTADDR=name]<br>[,IOREG=(r)<br>[,MODNAME=name]<br>[,RECSIZE={80\|n}]<br>[,SECADDR=SYSnnn]<br>[,SEPASMB=YES]<br>[,SORTMDE={ON\|OFF}] |

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 Macro').

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 27 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

| Name | Operation | Operand |
|---|---|---|
| [name] | DTFMT | BLKSIZE=n<br>,DEVADDR=SYSxxx<br>,EOFADDR=name<br>,FILABL={NO\|STD\|NSTD}<br>,IOAREA1={name\|(r)}<br>[,ASCII=YES]<br>[,BUFOFF={0\|n}]<br>[,CKPTREC=YES]<br>[,ERREXT=YES]<br>[,ERROPT={IGNORE\|SKIP\|name}]<br>[,HDRINFO=YES]<br>[,IOAREA2={name\|(r)}]<br>[,IOREG=(r)]<br>[,LABADDR=name]<br>[,LENCHK=YES]<br>[,READ={FORWARD\|BACK}]<br>[,RECFORM=format]<br>[,RECSIZE={n\|(r)}]<br>[,REWIND={UNLOAD\|NORWD}]<br>[,SEPASMB=YES]<br>[,TPMARK={YES\|NO}]<br>[,TYPEFLE={INPUT\|OUTPUT\|WORK}]<br>[,VARBLD=(r)]<br>[,WLRERR=name]<br>[,WORKA=YES] |

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). 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...'.
- 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 |
| 4 | If TYPEFLE=OUTPUT and RECFORM=VARUNB or VARBLK. 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.

  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 non-recoverable 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). ERROPT=name is required if you specify ERREXT=YES 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 (non-recoverable 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

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 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 nonstandard labels. 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 279.

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.

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}
A tapemark is normally written for an output file if non-standard labels are specified (FILABL=NSTD). If no tapemark is desired, specify TPMARK=NO. 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.

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.

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. 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

| Name | Operation | Operand |
|---|---|---|
| [name] | DTFOR | COREXIT=name<br>,DEVADDR=SYSxxx<br>,EOFADDR=name<br>,IOAREA1=name<br>[,BLKFAC=n]<br>[,BLKSIZE={38\|n}]<br>[,CONTROL=YES]<br>[,DEVICE=xxxxx]<br>[,HEADER=YES]<br>[,HPRMTY=YES]<br>[,IOAREA2=name]<br>[,IOREG=(r)]<br>[,MODNAME=name]<br>[,RECFORM={FIXUNB\|FIXBLK\|UNDEF}]<br>[,RECSIZE={n\|(r)}]<br>[,SEPASMB=YES]<br>[,WORKA=YES] |

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. 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, a non-recovery 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 a non-recoverable 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 a non-recoverable 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 a non-recoverable 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:

| Counter and Address | Contents |
|---|---|
| 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). |
| 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 executing 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).

All the previous 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. The binary contents of the counters should be converted 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.

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

| | | |
|---|---|---|
| 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 executing 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).

All the previous 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. The binary contents of the counters should be converted 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.

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 pro-

vides 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

| Name | Operation | Operand |
|---|---|---|
| [name] | DTFPH | TYPEFLE={INPUT\|OUTPUT}<br>[,ASCII=YES]<br>[,CISIZE=n]<br>[,CCWADDR=name]<br>[,DEVADDR=SYSxxx]<br>[,DEVICE=xxxx]<br>[,EOXPTR=name]<br>[,HDRINFO=YES]<br>[,LABADDR=name]<br>[,MOUNTED={ALL\|SINGLE}]<br>[,XTNTXIT=name] |

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 15 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 15. Operands to Define a Checkpoint File on Disk

ASCII=YES
: This operand is required to process ASCII tape files (see Appendix B). 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 Macro'). When processing ASCII tape files, the only valid specification is a programmer logical unit (that is, SYSnnn).

DEVICE=xxxx

Code the proper device identification, which may be one of the following:

TAPE If the file resides on tape mounted on an IBM 8809 or any IBM 2400/3400-series tape unit. 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 specifications for compatibility reasons:

| | |
|---|---|
| 2311 | 3340 |
| 2314 | 3350 |
| 3330 | |

EOXPTR=name

This operand is valid only if TYPEFLE=OUTPUT and MOUNTED=SINGLE is specified.

The operand specifies 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.

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.

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 279.

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.

ALL is specified 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.

SINGLE is specified 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).<br><br>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.

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

| Name | Operation | Operand |
|---|---|---|
| [name] | DTFPR | DEVADDR=SYSxxx<br>,IOAREA1=name<br>[,ASOCFLE=filename]<br>[,BLKSIZE=n]<br>[,CONTROL=YES]<br>[,CTLCHR={YES\|ASA}]<br>[,DEVICE=nnnn]<br>[,ERROPT={RETRY\|IGNORE\|name}]<br>[,FUNC=xxxx]<br>[,IOAREA2=name]<br>[,IOREG=(r)]<br>[,MODNAME=name]<br>[,PRINTOV=YES]<br>[,RDONLY=YES]<br>[,RECFORM={FIXUNB\|VARUNB\|UNDEF}]<br>[,RECSIZE=(r)]<br>[,SEPASMB=YES]<br>[,STLIST=YES]<br>[,TRC=YES]<br>[,UCS={ON\|OFF}]<br>[,WORKA=YES] |

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 2560, 3525, 5424, or 5425. For a discussion of associated files see the IBM publication Application Programming: 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 16 on page 125 defines the file name specified by the ASOCFLE operand for each of the associated DTFs.

| Your FUNC= Operand | Filename Specification in ASOCFLE=Operand of | | |
|---|---|---|---|
| | Read DTFCD | Punch DTFCD | Print DTFPR |
| FUNC=RPW | Filename of Punch DTFCD | Filename of Print DTFPR | Filename of Read DTFCD |
| FUNC=PW | | Filename of Print DTFPR | Filename of Punch DTFCD |
| FUNC=RW | Filename of Print DTFPR | | Filename of Read DTFCD |

Examples:

1. If FUNC=PW is specified, then:

   a. Specify the file name of the print DTFPR in the ASOCFLE operand of the punch DTFCD, and
   b. Specify the file name of the punch DTFCD in the ASOCFLE operand of the print DTFPR.

2. If FUNC=RPW is specified, then:

   a. Specify the file name of the punch DTFCD in the ASOCFLE operand of the read DTFCD, and
   b. Specify the file name of the print DTFPR in the ASOCFLE operand of the punch DTFCD, and
   c. Specify the file name of the read DTFCD in the ASOCFLE operand of the print DTFPR.

Figure 16. ASOCFLE Operand Usage with Print Associated Files

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 126.

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.
2. The control character of your record is X'5A' to indicate that this is a composed page data stream (CPDS) record.

| IBM Device | Maximum Length (in bytes)[1] | Assumed Length (in bytes)[2] |
|---|---|---|
| PRT1 | 168 | 121 |
| 1403-2, -3, -8, -9, | 132 | 121 |
| 1403-6, -7 | 120[3] | 121 |
| 1443 | 144 | 121 |
| 2560 | 384 | 64 |
| 3203 | 132 | 121 |
| 3525 | 64 | 64 |
| 3800/3200 | 384 (without TRC)[4] | 136 (without TRC)[4] |
| 5424/5425 | 128 | 96 |

[1] RECFORM is FIXUNB or UNDEF and operand CTLCHR is not specified.
[2] The operand BLKSIZE=n is omitted.
[3] Highest print position of the device.
[4] For a 3800, the maximum length is 385 if TRC=YES is used, and the assumed length is 137.

**Notes:**
1. If CTRCHR=YES/ASA is specified, add one byte to the maximum value which can be specified.
2. If RECFORM=VARUNB is specified add four bytes to the maximum value which can be specified.

Figure 17. Maximum and Assumed Lengths for the IOAREA1

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.

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.

This operand is not allowed for the IBM 2560, 5424, or 5425.

CTLCHR={YES|ASA}

Specify this operand if first-character control is used. CTLCHR=YES specifies the S/370 character set (see Appendix A for a list of codes). CTLCHR=ASA specifies the American National Standards Institute, Inc. character set.

If this operand is specified, omit CONTROL. This operand must not be specified for a file on the IBM 2560, 5424, or 5425.

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 2560, 3525, 5424, or 5425.

DEVICE=nnnn

This operand specifies the type of IBM device used for the file. Specify one of the following type codes:

| | | |
|---|---|---|
| PRT1 | 2560S | 3800 |
| 1403 | 3203 | 5425P |
| 1443 | 3211 | 5425S |
| 2560P | 3525 | |

The 'P' and 'S' included with 2560 and 5425 specify primary or secondary input hoppers. Specify 5425P and 5425S for the primary and secondary hoppers, respectively, for a file on an IBM 5424.

PRT1 refers to a 3211 or 3211-compatible IBM printer as listed in the IBM publication System Control Statements; 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:

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 17); 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 14), 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.

To continue processing at the end of the routine, return to IOCS by branching to the address in register 14.

FUNC={W[T]|RW[T]|RPW[T]|PW|[T]}
This operand specifies the type of file to be processed by the IBM 2560, 3525, or 5424/5425.

W indicates print, R indicates read, P indicates punch, and T (for the 3525 only) indicates an optional 2-line printer.

RW[T], RPW[T], and PW[T] are used, together with the ASOCFLE operand, to specify associated files; when one of these spec-

ifications (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. This operand is not allowed for a file on the IBM 2560, 5424, or 5425.

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 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 each 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

| Name | Operation | Operand |
|---|---|---|
| [name] | DTFSD | BLKSIZE={n\|MAX}<br>,EOFADDR=name<br>[,CISIZE=n]<br>[,DELETFL=NO]<br>[,DEVADDR=SYSxxx]<br>[,EOXPTR=name]<br>[,ERROPT={IGNORE\|SKIP\|name}]<br>[,FEOVD=YES]<br>[,HOLD=YES]<br>[,IOAREA1=name]<br>[,IOAREA2=name]<br>[,IOREG=(r)]<br>[,LABADDR=name]<br>[,PWRITE=YES]<br>[,RECFORM=format]<br>[,RECSIZE={n\|(r)}]<br>[,SEPASMB=YES]<br>[,TRUNCS=YES]<br>[,TYPEFLE={INPUT\|OUTPUT\|WORK}]<br>[,UPDATE=YES]<br>[,VARBLD=(r)]<br>[,VERIFY=YES]<br>[,WLRERR=name]<br>[,WORKA=YES] |

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 IBM publication Using the VSE/VSAM Space Management for SAM Feature.

BLKSIZE={n|MAX}

For 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 32,761 (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 32,761 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 32,761.

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 either 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 32,768.

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).

EOXPTR=name
: This operand specifies 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.

  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).

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 unrecoverable. Once the error is considered unrecoverable, the job is terminated unless the ERROPT operand is specified. 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.

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 the track hold function was specified at system generation time and if it is employed when a data input file or a work file is referenced for updating.

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 LBL 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. For more information about the handling of user-standard labels, see the sections "Processing of User Labels" on page 279.

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)}
: For fixed-length blocked records, RECSIZE is required. It specifies the number of characters in each record.

Register notation must be used when processing spanned or undefined records. When processing undefined records and variable-length spanned records, RECSIZE is required for output files and is optional for input files. The operand is invalid for work files. It specifies a general register (any one of 2 to 12) that contains the length of the record. On output, you must load the length of each record into the designated register before issuing a PUT macro. If specified for input, IOCS provides the length of the record transferred to virtual storage.

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}

Use this operand to indicate whether the file is an input or an output or a work file. If INPUT or OUTPUT is specified, the GET or PUT macros, respectively, must be used. If WORK is specified, the READ and WRITE, NOTE and POINTx, and CHECK macros must be used, and RECFORM must be either FIXUNB or UNDEF. If the operand is omitted, INPUT is assumed.

UPDATE=YES

This operand must be included if the disk input or work 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.

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.

  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.

  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

| Name | Operation | Operand |
|---|---|---|
| [name] | DTL | NAME=resourcename<br>[,CONTROL={E\|S}]<br>[,KEEP={NO\|YES}]<br>[,LOCKOPT={1\|2\|4}]<br>[,OWNER={TASK\|PARTITION}]<br>[,SCOPE={INT\|EXT}]<br>[,VOLID=volume-id] |

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 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. 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

| Name | Operation | Operand |
|---|---|---|
| [name] | DUMODFx | ERREXT=YES,ERROPT=YES<br>[,RDONLY=YES]<br>[,SEPASMB=YES] |

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 3-character prefix (IJN) and continues with a 5-character field corresponding to the options permitted in the generation of the module, as shown below.

DUMODFx name = IJNDabcd

a = I DUMODFI
  = O DUMODFO

b = C ERROPT=YES and ERREXT=YES
  = E ERROPT=YES
  = Z Neither is specified

c = Z Always

d = Y RDONLY=YES
  = Z RDONLY 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
          O E   Z
            Z
```

+ Subsetting/supersetting permitted
* No subsetting/supersetting permitted

## DUMP (Dump Request) Macro

| Name | Operation | Operand |
|---|---|---|
| [name] | DUMP | [RC={0\|n\|(15)}] |

This macro provides a hexadecimal dump of the following:

- The contents of the entire supervisor area and the used part of 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, which can be assigned to a printer, a disk, or a 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 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, a return code of 0 is assumed.

## ENDFL (End File Load Mode) Macro

| Name | Operation | Operand |
|---|---|---|
| [name] | ENDFL | {filename\|(0)} |

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

| Name | Operation | Operand |
|---|---|---|
| [name] | ENQ | {rcbname\|(0)} |

A task protects a resource by issuing an ENQ (enqueue) macro. When the RCB, (identified by rcbname) is enqueued, the task requesting the resource is either queued and executed, 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. If DEQ appears in the abnormal termination routine, 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 for a resource that is held by the main task, since a deadlock may occur.

## EOJ (End of Job) Macro

| Name | Operation | Operand |
|---|---|---|
| [name] | EOJ | [RC={0\|n\|(15)}] |

Issue the macro in the main task or in the only program 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, all abnormal termination exits (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.

  If the operand is omitted, a return code of 0 is assumed.

## ERET (Error-Handling Return) Macro

| Name | Operation | Operand |
|---|---|---|
| [name] | ERET | {SKIP\|IGNORE\|RETRY} |

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

| Name | Operation | Operand |
|---|---|---|
| [name] | ESETL | {filename\|(1)} |

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

| Name | Operation | Operand |
|---|---|---|
| [name] | EXCP | {blockname\|(1)}[,REAL] |

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 of such blocks 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.

blockname|(1)
Is the virtual address of the control block established for the device. It can be given as a symbol or in register notation.

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. In 370 mode 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 and, in 370 mode, also the IDA bit must be set to zero for these commands.

## EXIT (Return from Exit Routine) Macro

| Name | Operation | Operand |
|---|---|---|
| [name] | EXIT | {AB\|IT\|MR\|OC\|PC\|TT} |

The macro is used to return control from your exit control or MR routine to the instruction in your interrupted program immediately after the instruction where the interruption occurred. For AB, control is returned to the instruction following the EXIT AB macro. Your routine is specified in the STXIT macro (for MR, in the DTFMR 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.

MR Exit from your stacker selection routine (MICR document processing) to the supervisor.

OC Exit from your routine which handles the operator attention interrupt of your main task.

PC Exit from your program check routine.

TT Exit from your task timer routine.

Issue the EXIT macro only in the corresponding (AB, IT, MR, OC, PC, TT) routine; an illegal SVC may occur if this rule is not observed.

Information about the save area and the interrupt status is given in the IBM publication Service Aids.

For AB, the cancel condition and ABEND indication of the affected task are reset. The EXIT AB macro may be used only in main tasks. In a subtask, it would result in an illegal SVC. Ensure that the abnormal termination condition has been cleared up by your abnormal task termination routine before you use the EXIT AB macro.

For IT, OC, PC, and TT, the interrupt status information and registers are restored from the save area; thus, the save area contents are not over-written.

## EXTRACT (Extract Control Information) Macro

To display partition boundaries:

| Name | Operation | Operand |
|---|---|---|
| [name] | EXTRACT | ID=BDY<br>,AREA={name1\|(S,name1)\|(r1)}<br>,LEN={1\|length\|(r2)}<br>[,MFG={name3\|(r3)}]<br>[,MODE={T\|P}]<br>[,PID={name4\|(S,name4)\|(r4)}] |

To display unit information:

| Name | Operation | Operand |
|---|---|---|
| [name] | EXTRACT | ID=PUB<br>,AREA={name1\|(S,name1)\|(r1)}<br>,LEN={1\|length\|(r2)}<br>[,DISP={name6\|(S,name6)\|(r6)}]<br>[,MFG={name3\|(r3)}]<br>[,PID={name4\|(S,name4)\|(r4)}]<br>[,SEL={name5\|(S,name5)\|(r5)}] |

The macro may be used to retrieve and display partition boundaries or unit information (from the PUB table). The retrieved information can be interpreted with the help of the MAPBDY, MAPBDYVR, and IJBPUB macros.

ID={BDY|PUB}
: Specifies the information to be retrieved. BDY extracts partition boundaries, whereas PUB displays unit information.

AREA={name1|(S,name1)|(r1)}
: Specifies the address of the area where the extracted information is to be stored.

LEN={1|length|(r2)}
: Specifies the length of the area as an integer, a self-defining term, or as a value in a register. The default length is 1 byte.

DISP={name6|(S,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|P}
MODE=T indicates that the temporary boundaries of the issuing partition are to be returned. Do not specify the PID operand together with MODE=T; a snapshot of any other partition's temporary boundaries is unreliable.

If your program runs in real mode, the boundaries of the real partition (which, in ECPS:VSE mode, is contained in the corresponding virtual partition) are returned.

MODE=P indicates that the permanent boundaries 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.

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.

4 The partition specified is not supported in the system.

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.

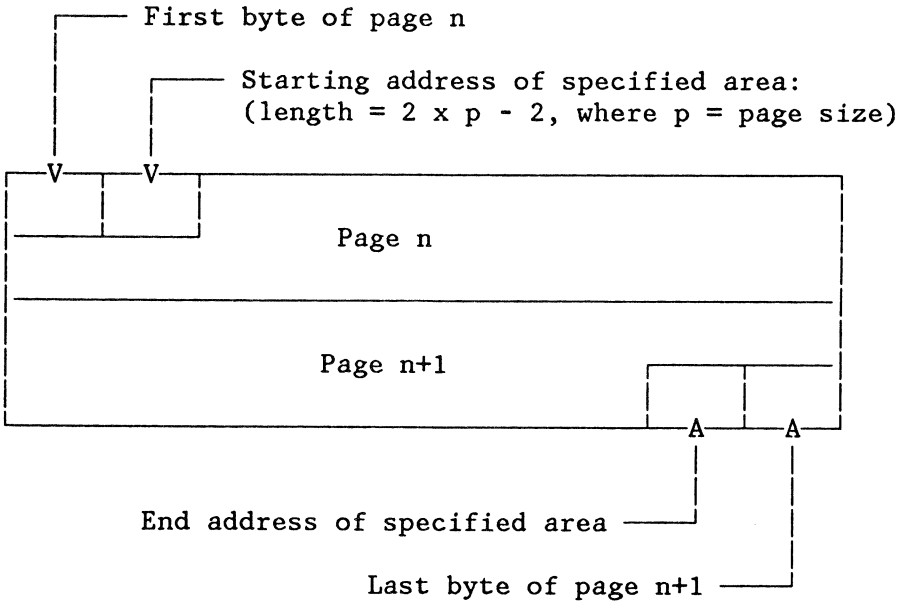
## FCEPGOUT (Force Page Out) Macro

You can code the macro in either of the following two formats:

| Name | Operation | Operand |
|---|---|---|
| [name] | FCEPGOUT | beginaddr,endaddr<br>[,beginaddr,endaddr] |

| Name | Operation | Operand |
|---|---|---|
| [name] | FCEPGOUT | {listname\|(1)} |

The macro causes a certain area in processor (real) storage to be scheduled for page-out. 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.

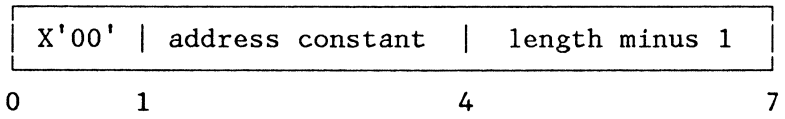



If your supervisor was generated with VM=YES (in the SUPVR generation macro), execution of the macro results in a null operation; the return code is set to zero.

beginaddr
: Points to the first location of the area to be paged out.

endaddr
Points to the last location of the area to be paged out.

listname|(1)
Is the symbolic name of a list of consecutive 8-byte entries as shown below.




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.

A non-zero byte following an entry indicates the end of the list. Register notation may also be used.

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 pending 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.

A page-handling request (page fault, temporary fix, PFIX) for at least one of the requested pages is pending (caused by asynchronous processing within a partition). The FCEPGOUT request has not been executed for those pages which are involved in a page page-handling request

16 The list of areas that are to be paged out is not completely in the requesting program's partition. The request is ignored.

Any combination of return codes 0, 2, 4, and 8 is possible.

## FEOV (Force End of Volume) Macro

| Name | Operation | Operand |
|---|---|---|
| [name] | FEOV | {filename\|(1)} |

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 macro.

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

| Name | Operation | Operand |
|---|---|---|
| [name] | FEOVD | {filename\|(1)} |

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

| Name | Operation | Operand |
|---|---|---|
| [name] | FETCH | {phasename\|(S,address)\|(1)}<br>[,entrypoint\|(S,entrypoint)\|(0)]<br>[,LIST={listname\|(S,listname)\|(r1)}]<br>[,SYS=YES]<br>[,DE={NO\|YES\|VSE[FORM]}]<br>[,MFG={area\|(S,area)\|(r2)}]<br>[,RET={NO\|YES}] |

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 the IBM publication System Management Guide.

phasename|(S,address)|(1)
: For phasename specify the name of the required phase. 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[FORM], 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. If this operand is not specified or invalid, 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.

If LIST is specified, a specification of YES or VSE[FORM] 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[FORM]}
By specifying DE=YES or DE=VSE[FORM], 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[FORM] 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 Version 2).

DE=VSE[FORM]
Indicates a 40-byte directory entry in the new (Release 2.1.0) librarian format. VSE is a short form of VSEFORM.

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:

| Bytes | |
|---|---|
| 0 | CL8'phasenam' |
| 8 | XL3'0' |
| 11 | XL1' No. of halfwords following ' |
| 12 | XL26'0' |

For DE=VSE[FORM]:

| Bytes | |
|---|---|
| 0 | CL8'phasenam' |
| 8 | XL3'FF' 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 Code 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 recataloged.

8 Unrecoverable 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.

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 loadpoint and partition start address
Difference between loadpoint and entrypoint

28 Partition (or logical transient area) is too small.

## FREE (Free Disk Area) Macro

| Name | Operation | Operand |
|---|---|---|
| [name] | FREE | {filename\|(1)} |

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 the IBM publication Application Programming: 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

| Name | Operation | Operand |
|---|---|---|
| [name] | FREEVIS | [ADDRESS={name1\|(1)}]<br>[,LENGTH={name2\|(0)}]<br>[,SPID={name3\|(1)}]<br>[,SVA={YES\|NO}] |

The macro releases a block (or blocks) 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.

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.

If the specified length is not a multiple of 128 or 16, respectively, it is rounded to the next higher integral multiple of 128 or 16.

SPID={name3|(1)}
Specifies that the subpool created by the SPID operand of the GETVIS macro is to be freed. The name of the subpool is specified in an eight-byte field addressed either by name3 or by a register.

SVA={YES|NO}
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.

Return Codes in Register 15

0 FREEVIS completed successfully.

4 The size of the (real) partition GETVIS area is OK.

8 The specified length is smaller than zero.

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 if the GETVIS area is in the SVA.

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.

## GENDTL (Generate the DTL Block) Macro

| Name | Operation | Operand |
|---|---|---|
| [name] | GENDTL | [ADDR={name1\|(S,name1)\|(r1)}]<br>[,CONTROL={E\|S}]<br>[,KEEP={NO\|YES}]<br>[,LENGTH={NO\|YES}]<br>[,LOCKOPT={1\|2\|4}]<br>[,NAME={name2\|(S,name2)\|(r2)}]<br>[,OWNER={TASK\|PARTITION}]<br>[,SCOPE={INT\|EXT}]<br>[,VOLID={name3\|(S,name3)\|(r3)}] |

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 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
'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 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. 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

| Name | Operation | Operand |
|---|---|---|
| [name] | GENIORB | CCW={name1\|(S,name1)\|(r1)}<br>,{DEVICE=SYSxxx\|<br>LOGUNIT={name2\|(S,name2)\|(r2)}}<br>[,ADDRESS={name3\|(S,name3)\|(r3)}]<br>[,LENGTH=fieldlength]<br>[,ECB={name4\|(S,name4)\|(r4)}]<br>[,ERREXIT={name5\|(S,name5)\|(r5)}]<br>[,FIXLIST={name6\|(S,name6)\|(r6)}]<br>[,FIXFLAG=(option,...)]<br>[,IOFLAG=(option,...)] |

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 18 on page 172. The IORB is an alternative to the CCB; instead of specifying a CCB in the EXCP macro, the address of an IORB is given.

The IORB requires the specification of areas to be page-fixed for the I/O operation. Such areas include the IORB and the channel pro-

grams themselves and all input/output areas. Specifying those areas frees the page-fixing routines from having to scan the channel programs to determine which areas are to be fixed.

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 12 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 descripton of the ECB operand of the ATTACH macro on page 8.

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).

FIXLIST=, FIXFLAG=, IOFLAG=
For a description of these operands, see the IORB macro.

## GENL (Generate Directory List) Macro

| Name | Operation | Operand |
|---|---|---|
| [name] | GENL | phasename1,phasename2,...<br>[,ADDRESS={area\|(S,area)\|(r1)}<br>,LENGTH=number]<br>[,ADDRESS={DYNAMIC\|DYN}<br>[,ERREXIT={addr\|(S,addr)\|(r2)}]]<br>[,DE={OLD\|VSE[FORM]}] |

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)},LENGTH=number
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.

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}[,ERREXIT={addr|(S,addr)|(r2)}]:
Coding ADDRESS=DYNAMIC (a short form, ADDRESS=DYN, is allowed) directs the system to acquire, through a GETVIS, as much dynamic storage as needed. In this case, the contents of registers 0, 1, 14, and 15 are over-written by execution of the macro.

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[FORM]}
Indicates the format of the directory entries to be generated. DE=OLD generates 38-byte directory entries in the old (VSE Version 1) format. DE=VSE[FORM] generates 40-byte directory entries in the new (VSE Version 2) librarian format. VSE is a short form of VSEFORM.

## GET (Get a Record) Macro

| Name | Operation | Operand |
|---|---|---|
| [name] | GET | {filename\|(1)}<br>[,workname\|,(0)] |

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.

In conjunction with optical reader input, GET can be used only to retrieve records from a journal tape on a 1287.

## GETIME (Get the Time) Macro

| Name | Operation | Operand |
|---|---|---|
| [name] | GETIME | [STANDARD\|BINARY\|TU\|MIC]<br>[,CLOCK=YES]<br>[,LOCAL\|,GMT]<br>[,MFG={area\|(S,area)\|(r)}] |

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, 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.

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. This value is transformed into time and date as defined by the other operands and any associated job control statements. On output, register 1 contains the time, and registers 14 and 15 the date (in the form mmddyy00 or ddmmyy00).

LOCAL|GMT
Specify LOCAL to obtain the local time or GMT if, in your program, you want to use Greenwich Mean Time.

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.

## GETVIS (Get Virtual Storage) Macro

| Name | Operation | Operand |
|---|---|---|
| [name] | GETVIS | [ADDRESS={name1\|(1)}]<br>[,LENGTH={name2\|(0)}]<br>[,PAGE={NO\|YES}]<br>[,PFIX={NO\|YES}]<br>[,POOL={NO\|YES}]<br>[,SPID={name3\|(1)}]<br>[,SVA={NO\|YES}]<br>[,TSKSUBP={NO\|YES}] |

The macro retrieves a block of virtual storage from the GETVIS area of your partition or of the SVA. 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.

  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.

PAGE={NO|YES}
: If you want the requested storage area to start on a page boundary, specify PAGE=YES. 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.

POOL={NO|YES}
If POOL=YES is specified, GETVIS starts searching for the requested storage area. To do so, GETVIS expects to find, in register 1, an address within the defined GETVIS area.

The implementation of this operand for Version 2 of VSE/Advanced Functions differs from this implementation for Version 1. The address returned by the macro in register 1 may be any address within the defined GETVIS area.

POOL=YES must not be specified together with the SPID operand or with TSKSUBP=YES.

SPID={name3|(1)}
This operand can be used to create a subpool of pages of virtual storage from the GETVIS area of your partition or of 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 two parts: the first six bytes contain the subpool name which you must supply, and the second two bytes must be initialized to zero when you first create the subpool. This two-byte field is set by the system. The entire eight-byte field must be used for subsequent references to the subpool. The subpool ID field is passed to the supervisor in register 1.

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 pertinent partition.

SVA=YES must not be specified together with 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 SPID, POOL or SVA.

Return Codes in Register 15

| | |
|---|---|
| 0 | GETVIS completed successfully. |
| 4 | The size of the (real) partition GETVIS area is OK. |
| 8 | The specified length is negative or exceeds the GETVIS area. |
| 12 | No more virtual storage is available in the GETVIS area. |
| 16 | The maximum number of subpools is exhausted. |
| 20 | Invalid GETVIS option. |
| 24 | Invalid subpool ID. |
| 32 | PFIX for an SVA subpool request failed. |

## IJBPUB (IJBPUB DSECT) Macro

| Name | Operation | Operand |
|---|---|---|
| [name] | IJBPUB | |

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.

## IORB (I/O Request Block Definition) Macro

Use one of the formats shown below:

| Name | Operation | Operand |
|---|---|---|
| [name] | IORB | DSECT=YES |

| Name | Operation | Operand |
|---|---|---|
| [name] | IORB | CCW=name1,DEVICE=SYSxxx<br>[,ECB=name2]<br>[,FIXLIST=name3]<br>[,FIXFLAG=(option,...)]<br>[,IOFLAG=(option,...)] |

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 18.

| Offset (In Hex) | Length (In Hex) | Contents |
|---|---|---|
| 0 | 2 | Residual count (same as for a CCB; see Figure 2 on page 12). |
| 2 | 2 | Transmission information (same as for a CCB; see Figure 2 on page 12). |
| 4 | 2 | CSW status bits (same as for a CCB; see Figure 2 on page 12). |
| 6 | 7 | Reserved. |
| 0D | 3 | CSW address in CSW (same as for a CCB; see Figure 2 on page 12). |
| 10 | 10 | Reserved. |

Figure 18. Layout and Contents of the I/O Request Block (IORB)

The block is an alternative to the CCB: instead of specifying a CCB in the EXCP macro, the address of an IORB is given. The IORB macro requires the specification of areas to be page-fixed for the I/O operation. Such areas include the IORB and the channel programs themselves and all input/output areas. Specifying those areas frees

the page-fixing routines from having to scan the channel programs to determine which areas are to be fixed.

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.

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.

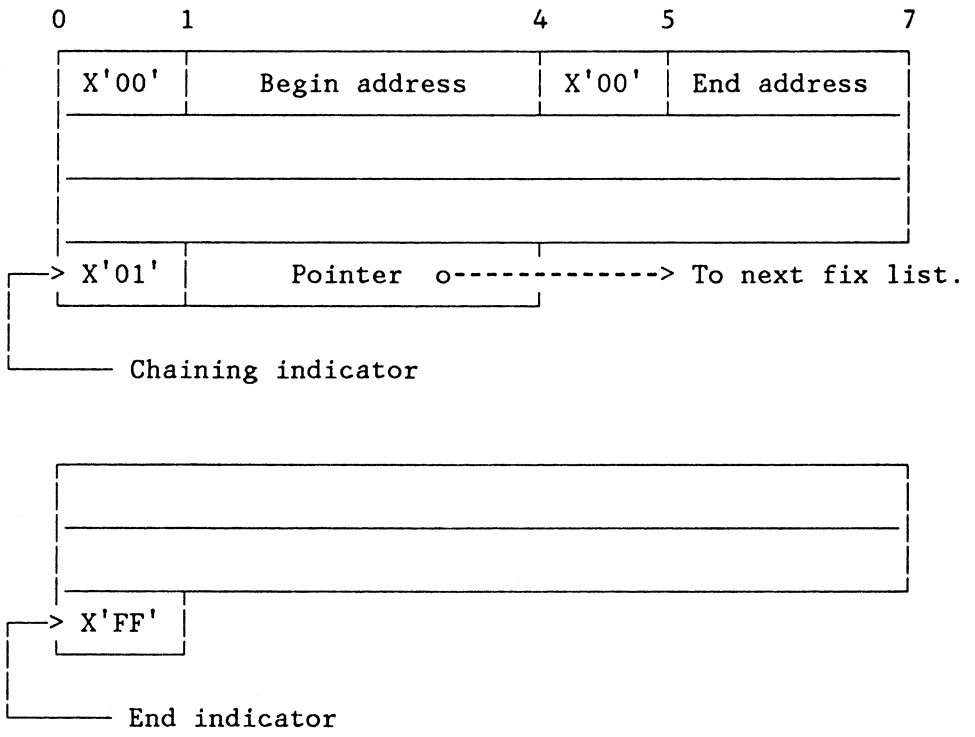
DEVICE=SYSxxx
: This operand specifies the logical unit for the actual I/O unit with which this IORB is associated.

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.

FIXLIST=name3
: This operand specifies the address of the first of two or more fix list parts or of the only fix list part. The FIXLIST operand is required unless FIXFLAG=(FIXED) is specified. Each fix list part consists of one or more 8-byte entries plus an end or chaining indicator as shown on page 174.

  In a fix list entry, begin address and end address are the addresses of the first and the last byte of an area which has to be fixed for the I/O request (begin address ≤ end address; entries with begin address > end address will be canceled with an 'invalid address' message). Each entry describes a storage area that is accessed by the channel during the I/O request; that is, an area containing the channel program, an input/output area, or the IORB.




Duplicate entries and entries describing overlapping storage areas are allowed. As a result, certain areas may be covered more than once by the fix list. The system will compress the fix list so that each page to be fixed for the channel program is covered only once. However, specifying FIXFLAG=(COMPRESSED) indicates that this service is not applicable or has already been done by the user.

FIXFLAG=(option,...)
This operand specifies a list of options which apply to the I/O fixing procedure. In 370 mode, this operand is ignored.

The options you can specify are:

COMPRESSED To indicate that the system does not have to compress the fix list. Use this option if the fix list is already compressed, that is: each page to be fixed for the I/O request is covered only once by the fix list.

FIXED To indicate that all areas which should be fixed for the I/O operation have already been fixed by the user, there is no fix list.

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 unrecoverable I/O error is to be accepted.

SKIPERP To indicate that error recovery by the system is to be skipped.

## ISMOD (Indexed Sequential I/O Module Definition) Macro

| Name | Operation | Operand |
|---|---|---|
| [name] | ISMOD | IOROUT={LOAD\|ADD\|RETRVE\|ADDRTR}<br>[,CORDATA=YES]<br>[,CORINDX=YES]<br>[,ERREXT=YES]<br>[,HOLD=YES]<br>[,IOAREA2=YES]<br>[,RDONLY=YES]<br>[,RECFORM={FIXUNB\|FIXBLK\|BOTH}]<br>[,RPS=SVA]<br>[,SEPASMB=YES]<br>[,TYPEFLE={RANDOM\|SEQNTL\|RANSEQ}] |

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 unrecoverable 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

```
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.

| | | |

  - 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

| Name | Operation | Operand |
|---|---|---|
| [name] | JOBCOM | FUNCT={PUTCOM\|GETCOM},<br>AREA={address\|(r1)},<br>LENGTH={length\|(r2)} |

The macro provides for communication between jobs or job steps in a partition. 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

| Name | Operation | Operand |
|---|---|---|
| [name] | LBRET | {1\|2\|3} |

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.

## LFCB (Load Forms Control Buffer) Macro

| Name | Operation | Operand |
|---|---|---|
| [name] | LFCB | SYSxxx,phasename<br>[,NULMSG]<br>[,FORMS=xxxx]<br>[,LPI=n] |

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 the IBM publication 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. However, for an IBM 3800, register 15 contains 0, although the macro was not executed. If the operation fails, register 15 contains one of the return codes listed below; in this case the FCB retains its original contents. 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 is invalid for the printer assigned to the specified logical unit.

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.

## LITE (Pocket-Light Control) Macro

| Name | Operation | Operand |
|---|---|---|
| [name] | LITE | {filename\|(1)}<br>[,light-switches\|,(0)] |

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 27). 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 19.

| 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 19. Bit Configuration of the Pocket-Light Switch Area

## LOAD (Load a Phase) Macro

| Name | Operation | Operand |
|---|---|---|
| [name] | LOAD | {phasename\|(S,address)\|(1)}<br>[,loadpoint\|(S,loadpoint)\|(0)]<br>[,LIST={listname\|(S,listname)\|(r1)}]<br>[,SYS=YES]<br>[,DE={NO\|YES\|VSE[FORM]}]<br>[,TXT=NO]<br>[,MFG={area\|(S,area)\|(r2)}]<br>[,RET={NO\|YES}] |

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 phase. After execution of the macro, the entry-point address of the called phase is returned to you in register 1. For a non-

relocatable phase, this address 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.

phasename|(S,address)|(1)
: For phasename specify the name of the required phase. 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|VSE[FORM], the operand has a different meaning; refer to the discussion of the DE operand.

loadpoint|(S,loadpoint)|(0)
: If loadpoint is provided, the load-point address specified to the linkage editor is overridden, and the phase is loaded at the specified address. 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.

  If LIST is specified, DE=YES|VSE[FORM] 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[FORM]} By specifying DE=YES or DE=VSE[FORM] 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[FORM] 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 (prior to Version 2) librarian format is to be generated.

DE=VSE[FORM]
Indicates that a 40-byte directory entry in the new (Version 2) librarian format is to be generated. VSE is a short form of VSEFORM.

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) and have the following format:

For DE=YES:

| Bytes | | |
|---|---|---|
| 0 | CL8'phasenam' | |
| 8 | XL3'0' | |
| 11 | XL1' | No. of halfwords following' |
| 12 | XL26'0' | |

For DE=VSE[FORM]:

| Bytes | | |
|---|---|---|
| 0 | CL8'phasenam' | |
| 8 | XL3'FF' | ID for new format |
| 11 | XL1' | 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'.

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.

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 recataloged.

8 Unrecoverable 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.

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.

28 Partition (or logical transient area) is too small.

## LOCK (Lock a Resource) Macro

| Name | Operation | Operand |
|---|---|---|
| [name] | LOCK | {name\|(S,name)\|(r)}<br>[,FAIL={RETURN\|WAITC\|WAIT}] |

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}
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.

WAIT or WAITC cannot be specified if the resource is defined with OWNER=PARTITION.

Figure 20 on page 192 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 |

G = The LOCK request is granted (return code 0).
I = 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 own system. Access is denied (return code 4) if the resource is exclusively held by the other system.

Figure 20. 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 21 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 21.

| Return Code | | System Action for LOCK FAIL= | | |
|---|---|---|---|---|
| Hex. | Dec. | RETURN | WAITC | WAIT |
| 00 | 0 | Return | Return | Return |
| 04 | 4 | Return | Wait | Wait |
| 08 | 8 | Return | Return | Wait |
| 0C | 12 | Return | Return | Cancel |
| 10 | 16 | Return | Return | Cancel |
| 14 | 20 | Return | Return | Cancel |
| 18 | 24 | Return | Return | Cancel |
| 1C | 28 | Return | Return | Wait |
| 20 | 32 | Return | Return | Cancel |
| 24 | 36 | Return | Return | Cancel |

Figure 21. 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.

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 on-line.

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

| Name | Operation | Operand |
|---|---|---|
| [name] | MAPBDY | [DSECT={NO\|YES}] |

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, in-line code is generated.

For the layout and a description of the DSECT fields, see Figure 22 on page 195.

| Field Name | No. of Bytes | Description |
|---|---|---|
| PBEGIN | 4 | Partition start address, corresponding to problem program save area address (field PIB SAVE). |
| PENDLOG | 4 | Logical end of partition (last addressable byte, GETVIS area excluded), corresponding to field PPEND in the partition communication region. |
| PGEND | 4 | Physical end of partition (last addressable byte, GETVIS area included). |
| PFIXLMT | 4 | PFIX limit (K-bytes) or zero (real mode). |
| PFIXCNT | 4 | PFIX count (number of PFIXed pages). |

Figure 22. Layout of the MAPBDY-generated DSECT

## MAPBDYVR (Map Boundary Information) Macro

| Name | Operation | Operand |
|---|---|---|
| [name] | MAPBDYVR | [DSECT={NO\|YES}] |

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, in-line code is generated.

For the layout and a description of the DSECT fields, see Figure 23 on page 196.

| Field Name | No. of Bytes | Description |
|---|---|---|
| VPBEGIN | 4 | Virtual partition start address (corresponding to the latest allocation). |
| VPEND | 4 | Virtual partition logical end address (last addressable byte, GETVIS area excluded). |
| VPGEND | 4 | Virtual partition physical end address (last addressable byte, GETVIS area included, corresponding to latest allocation). |
| RPBEGIN | 4 | Real partition start address (corresponding to latest allocation). |
| RPEND | 4 | Real partition end address (last addressable byte, corresponding to latest allocation). |

Figure 23. Layout of the MAPBDYVR-generated DSECT

## MAPSSID (Map for SUBSID) Macro

| Name | Operation | Operand |
|---|---|---|
| [name] | MAPSSID | |

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 24 on page 197.

| Offset (In Hex) | Length (In Bytes) | Content/Description |
|---|---|---|
| **Fixed part:** | | |
| 00 | 2 | Always zero. |
| 02 | 4 | Character string: SUP. |
| 06 | 1 | Version number. |
| 07 | 1 | Release number. |
| 08 | 1 | Modification level. |
| 09 | 1 | Length of variable part (a maximum of 24 bytes) |
| **Variable part:** | | |
| 0A | 1 | Flag byte 1:<br>X'80' 370-mode supervisor.<br>X'40' ECPS:VSE mode supervisor.<br>X'20' CKD support available.<br>X'10' FBA support available.<br>X'08' 3800 support available.<br>X'04' Relocating channels (ECPS:VSE mode only).<br>X'02' MODE=VM supervisor.<br>X'01' MODE=VM supervisor active under VM control. |
| 0B | 1 | Flag byte 2:<br>X'80' Always 1.<br>X'40' 4K page size |
| 0C | 1 | Flag byte 3:<br>X'80' Access-control support is available.<br>X'40' DASD sharing support is available.<br>X'20' For internal use. |
| 0D | 1 | Flag byte 4: Reserved. |
| 0E | 2 | Length of sublibrary search chain. |

Figure 24. Layout of MAPSSID-generated DSECT

## MODDTL (Modify DTL Block) Macro

| Name | Operation | Operand |
|---|---|---|
| [name] | MODDTL | ADDR={name1\|(S,name1)\|(r1)}<br>[,NAME={name2\|(S,name2)\|(r2)}]<br>[,CHANGE={ON\|OFF}]<br>[,CONTROL={E\|S}]<br>[,LOCKOPT={1\|2\|4}]<br>[,KEEP={NO\|YES}]<br>[,OWNER={TASK\|PARTITION}]<br>[,SCOPE={INT\|EXT}]<br>[,VOLID={name3\|(S,name3)\|(r3)}] |

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

| Name | Operation | Operand |
|---|---|---|
| [name] | MRMOD | [ADDRESS={SINGLE\|DUAL}]<br>[,BUFFERS=n]<br>[,SEPASMB=YES] |

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

| Name | Operation | Operand |
|---|---|---|
| [name] | MVCOM | to,length,{from\|(0)} |

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 the IBM publication Application Programming: 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

| Name | Operation | Operand |
|---|---|---|
| [name] | NOTE | {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. 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 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

| Name | Operation | Operand |
|---|---|---|
| [name] | OPEN | {filename1\|(r1)}<br>[,filename2\|,(r2)],... |

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-self-relocating 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 should not be attempted for the file; the results may be unpredictable.

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 pre-loaded into any of the registers 2 through 12. For OPEN, this address may be pre-loaded into register 0 or any of the registers 2 through 12.

## ORMOD (Optical Reader Input Module Definition) Macro

| Name | Operation | Operand |
|---|---|---|
| [name] | ORMOD | DEVICE={1287D\|1287T}<br>[,BLKFAC=YES]<br>[,CONTROL=YES]<br>[,IOAREA2=YES]<br>[,RECFORM={FIXUNB\|FIXBLK\|UNDEF}]<br>[,SEPASMB=YES]<br>[,WORKA=YES] |

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

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 two formats:

| Name | Operation | Operand |
|---|---|---|
| [name] | PAGEIN | beginaddr,endaddr<br>[,beginaddr,endaddr]...<br>[,ECB={ecbname\|(0)}] |

| Name | Operation | Operand |
|---|---|---|
| [name] | PAGEIN | {listname\|(1)}<br>[,ECB={ecbname\|(0)}] |

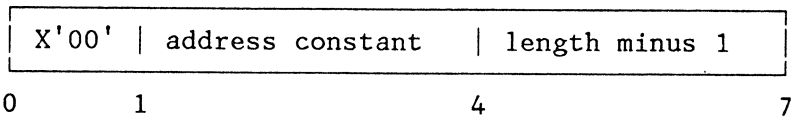
The macro causes specific areas to be brought into real storage before their contents are needed by the requesting program. If the requested area is already in real storage the attached page frame will get low priority for the next page-outs. This function, however, does not include any fixing, so that 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. On a system that was generated with VM=YES (in the SUPVR generation macro), 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. A non-zero byte following an entry indicates the end of the list.




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.

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. |
| 6-7 | Reserved. |

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

| Name | Operation | Operand |
|---|---|---|
| [name] | PDUMP | {address1\|(r1)},{address2\|(r2)}<br>[,MFG={area\|(S,area)\|(r3)}] |

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 over-written, but the CPU status is retained. Thus, PDUMP furnishes a dynamic dump (snapshot) useful for program checkout. Processing continues with your next instruction.

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 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 SYSLST is assigned to disk, processing a PDUMP macro requires you to issue an OPEN macro to any DTF that refers to SYSLST as output device. The OPEN macro updates the disk address maintained in the DTF table to agree with the address where the PDUMP output ends. If this OPEN is not issued, the address is not updated, and your program is canceled when it issues the next PUT.

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. 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 two formats:

| Name | Operation | Operand |
|---|---|---|
| [name] | PFIX | beginaddr,endaddr<br>[,beginaddr,endaddr]... |

| Name | Operation | Operand |
|---|---|---|
| [name] | PFIX | {listname\|(1)}<br>[,beginaddr,endaddr]... |

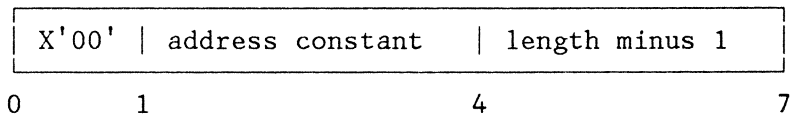
The macro causes specific pages to be brought into real storage and fixed in their page frames until they are released at some later time. The maximum number of pages that may be fixed at any one time is specified via the ALLOCR command. Each time a page is fixed a counter for that page is incremented. This counter may never exceed 255 for any page.

If your supervisor was generated with VM=YES (in the SUPVR generation macro), 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 fixed.

endaddr
Points to the last byte of the area to be fixed.

listname|(1)
Is the name of a list of consecutive 8-byte entries as shown below. A non-zero byte following an entry indicates the end of the list. Register notation may be used.




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.

Exceptional Conditions

- If a PFIX causes the count of fixes for a page to exceed 255, 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 ALLOCR 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.

## PFREE (Page-Free Request) Macro

You can code the macro in either of the following two formats:

| Name | Operation | Operand |
|---|---|---|
| [name] | PFREE | beginaddr,endaddr<br>[,beginaddr,endaddr]... |

| Name | Operation | Operand |
|---|---|---|
| [name] | PFREE | {listname\|(1)} |

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.

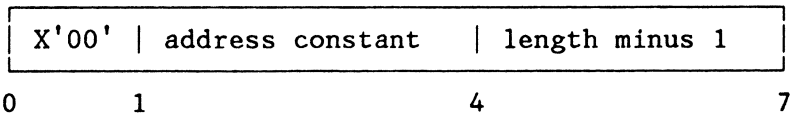
The PFREE macro decrements the counter of a specified page by 1. If a PFREE is issued for a page whose counter is 0, that PFREE is ignored.

If your supervisor was generated with VM=YES (in the SUPVR generation macro), 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 freed.

endaddr
: Points to the last byte of the area to be freed.

listname(1)
: Is the symbolic name of a list of consecutive 8-byte entries as shown below. A non-zero byte following an entry indicates the end of the list.




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.

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.

## POINTR (Point to Noted Record) Macro

| Name | Operation | Operand |
|---|---|---|
| [name] | POINTR | {filename\|(1)},{address\|(0)} |

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 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 POINTR.

## POINTS (Point to Start) Macro

| Name | Operation | Operand |
|---|---|---|
| [name] | POINTS | {filename\|(1)} |

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

| Name | Operation | Operand |
|---|---|---|
| [name] | POINTW | {filename\|(1)},{address\|(0)} |

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

| Name | Operation | Operand |
|---|---|---|
| [name] | POST | {ecbname\|(1)}<br>[,SAVE={savearea\|(0)}] |

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 8.

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.

Although specifying this operand saves time, 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. On the other hand, 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, and the highest priority task gets control.

## PRMOD (Printer Output Module Definition) Macro

| Name | Operation | Operand |
|---|---|---|
| [name] | PRMOD | [CONTROL=YES]<br>[,CTLCHR={YES\|ASA}]<br>[,DEVICE=xxxxx]<br>[,ERROPT=YES]<br>[,FUNC=xxx]<br>[,IOAREA2=YES]<br>[,PRINTOV=YES]<br>[,RDONLY=YES]<br>[,RECFORM={FIXUNB\|VARUNB\|UNDEF}]<br>[,SEPASMB=YES]<br>[,STLIST=YES]<br>[,TRC=YES]<br>[,WORKA=YES] |

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.

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.

The CONTROL operand is not allowed for a file on the IBM 2560, 5424, or 5425.

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.

CTLCHR must not be specified for a file on the IBM 2560, 5424, or 5425.

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 | 2560S | 3525 |
| 1443 | 3203 | 5425P |
| 2560P | 3211 | 5425S |

A device code of 3800 and PRT1 is accepted for compatibility reasons. The generated logic module is ignored, however.

The characters P and S at the end of 2560 and 5425 specify primary or secondary input hoppers. Regardless of what is specified, the generated module can handle DTFs specifying either hopper. Specify 5425P or 5425S (whichever applies) for a file on the IBM 5424.

Any DTF to be used with this module must have the same operand (except as just noted concerning the 'P' and 'S' specification for a file on the IBM 2560, 5424, or 5425).

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[T]|RW[T]|RPW[T]|PW[T]}
This operand specifies the type of file to be processed on your IBM 2560, 3525, 5424, or 5425. 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[T], RPW[T], and PW[T] 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.

This operand is not allowed for a file on the IBM 2560, 5424, or 5425.

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

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
  = U DEVICE=2560
  = V DEVICE=5425

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[T] or omitted
  = F PRINTOV=YES, DEVICE=3525, and FUNC=RW[T]
  = C PRINTOV=YES, DEVICE=3525, and FUNC=PW[T]
  = D PRINTOV=YES, DEVICE=3525, and FUNC=RPW[T]
  = Z Neither PRINTOV nor ERROPT is specified, and DEVICE is not a 3525
  = O PRINTOV=YES not specified, DEVICE=3525, and FUNC=W[T] or omitted
  = R PRINTOV=YES not specified, DEVICE=3525, and FUNC=RW[T]
  = S PRINTOV=YES not specified, DEVICE=3525, and FUNC=PW[T]
  = T PRINTOV=YES not specified, DEVICE=3525, and FUNC=RPW[T]
  = E ERROPT=YES and PRINTOV=YES is not specified
  = U FUNC=W or omitted and DEVICE=2560 or 5425
  = V FUNC=RW and DEVICE=2560 or 5425
  = W FUNC=PW and DEVICE=2560 or 5425

= X FUNC=RPW and DEVICE=2560 or 5425

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. No subsetting or supersetting of PRMOD names is allowed for the IBM 2560 or 5424/5425.

**Note:** 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 PRMOD logic modules are reassembled this way, it may interfere with subsetting or supersetting.

| I | J | D | F V U | A Y S T U V + C Z | U V W X + P Z + I O + F R + + C S + D T + B E | I Z | V W Y Z |
|---|---|---|---|---|---|---|---|
| * | * | * | * | * | * | * | * |

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

## PRTOV (Printer Overflow Control) Macro

| Name | Operation | Operand |
|---|---|---|
| [name] | PRTOV | {filename\|(1)},{9\|12}<br>[,routine-name\|,(0)] |

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

| Name | Operation | Operand |
|---|---|---|
| [name] | PUT | {filename\|(1)}<br>[,workname\|,(0)]<br>[,STLSP={controlfield\|(r1)}]<br>[,STLSK={controlfield\|(r2)}] |

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 |
| 1 | 7 |
| 2 | 6 |
| 3 | 5 |
| 4 | 4 |
| 5 | 3 |
| 6 | 2 |
| 7 | 1 |

The tape position 1 is the leftmost tape on the selective tape listing device.

**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

| Name | Operation | Operand |
|---|---|---|
| [name] | PUTR | {filename\|(1)}<br>[,{workname1\|(0)},<br>{workname2\|(2)}] |

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 3210 or 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 3210 or 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.

## RCB (Resource Control Block Definition) Macro

| Name | Operation | Operand |
|---|---|---|
| [name] | RCB | |

The RCB macro generates an 8-byte word-aligned resource control block (RCB); this block 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 format of the RCB and its use 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 | 0 | Another task is waiting to use the resource. |
| | 1-7 | Reserved. |
| 5-7 | ECB address of current resource owner. | |

## RDLNE (Read a Line) Macro

| Name | Operation | Operand |
|---|---|---|
| [name] | RDLNE | {filename\|(1)} |

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 off-line correction mode. RDLNE should be used in the COREXIT routine only, or else the line following the one in error will be read in on-line 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, he may enter the reject character 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

| Name | Operation | Operand |
|---|---|---|
| [name] | READ | {filename\|(1)}<br>{,SQ,{area\|(0)}[,length\|,(r1)\|,S]\|<br>,ID\|<br>,KEY\|<br>,OR,{name\|(r2)}\|<br>,DR,{name\|(r3)\|number,number}\|<br>,MR} |

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
Used only for 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 DTF filenameL field.

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

| Name | Operation | Operand |
|---|---|---|
| [name] | REALAD | {address\|(1)} |

In 370 mode, the macro returns the real address corresponding to a specified virtual address. If issued in ECPS:VSE mode, the macro returns the 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 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.

**Note:** The pages of a partition running in real mode are treated as if they were fixed.

## RELEASE (Release Logical Unit) Macro

| Name | Operation | Operand |
|---|---|---|
| [name] | RELEASE | (SYSnnn[,SYSnnn]...)<br>[,savearea] |

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 over-written.

  The macro expansion includes a unit table and loads the table's address into register 0. If the savearea 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 two formats:

| Name | Operation | Operand |
|---|---|---|
| [name] | RELPAG | beginaddr,endaddr<br>[,beginaddr,endaddr]... |

| Name | Operation | Operand |
|---|---|---|
| [name] | RELPAG | {listname\|(1)} |

The RELPAG macro causes the contents of one or more 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.

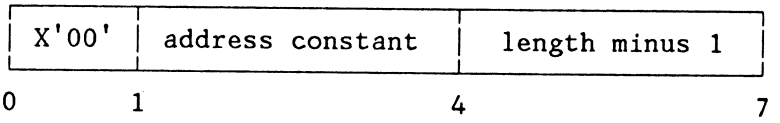
After the RELPAG macro has been executed for an area and a location in that area is referenced again during the current program execution, the related page is attached to a page frame which contains all zeros.

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 152.

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 symbolic name of a list of consecutive 8-byte entries as shown below.




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.

A non-zero byte following an entry indicates the end of the list.

Register notation may be used.

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.

A page handling request (page fault, temporary fix, PFIX) for at least one of the requested pages is pending (caused by asynchronous processing within a partition). The release request has not been executed for those pages which are involved in a page-handling request.

16 List of areas that are to be released is not completely in the requesting program's partition. The request is ignored.

Any combination of the return codes 2, 4, and 8 is possible.

## RELSE (Release a Block) Macro

| Name | Operation | Operand |
|---|---|---|
| [name] | RELSE | {filename\|(1)} |

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

| Name | Operation | Operand |
|---|---|---|
| [name] | RESCN | {filename\|(1)}<br>,(r1),(r2)[,n1][,n2] |

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.

**Notes:**

1. For a file on the IBM 1287 models 3 or 4 or the IBM 1288, this macro can be used only with READ BACKWARD commands.
2. If you use the macro with a READ FORWARD command, the input area is not cleared.
3. When unformatted fields are read from a file on the IBM 1288, the RESCN macro should not be used.

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 on-line 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

| Name | Operation | Operand |
|---|---|---|
| [name] | RETURN | (r1[,r2]) |

The RETURN macro restores the registers whose contents were saved and returns control to the calling program.

(r1[,r2])
: Specify the range of the registers to be reloaded 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.

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.

## RUNMODE (Run-Mode Indication) Macro

| Name | Operation | Operand |
|---|---|---|
| [name] | RUNMODE | |

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

| Name | Operation | Operand |
|---|---|---|
| [name] | SAVE | (r1[,r2]) |

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.

## SECTVAL (Sector-Value Calculation) Macro

| Name | Operation | Operand |
|---|---|---|
| [name] | SECTVAL | [DDKR={name1\|(0)}]<br>,DVCTYP=name2 |

The macro calculates the sector value of the address 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 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. If no operand is specified, register 0 is the default and should contain the necessary information. 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 number of data bytes used on the track, excluding standard R0 length up to the current record. Bit 0 of the first byte must be set on and the record field (r) or key byte (k) must be non-zero.

  If the k and r bytes are zero, the dd field is considered as a 16-bit number specifying the total number of bytes used so far on the track (except R0, but including all other overhead).

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.

DVCTYP=name2

The device type code must be supplied at name2. The following device type codes (the same as those contained in the DTF block) are valid for the devices as indicated:

| IBM Device | Code |
|---|---|
| 3330, Models 1 and 2 | X'04 |
| 3330, Model 11 | X'05 |
| 3340 | X'08', X'09', or X'0A' |
| 3350 | X'07 |
| 3375 | X'0B |
| 3380 | X'0C |

**Note:** The Version 1 implementation (option to supply the code in a register) is maintained to ensure program compatibility.

## SEOV (System End-of-Volume) Macro

| Name | Operation | Operand |
|---|---|---|
| [name] | SEOV | filename |

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

| Name | Operation | Operand |
|---|---|---|
| [name] | SETDEV | {filename\|(1)},{phasename\|(r)} |

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

| Name | Operation | Operand |
|---|---|---|
| [name] | SETFL | {filename\|(0)} |

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

| Name | Operation | Operand |
|---|---|---|
| [name] | SETIME | {timervalue\|(1)}<br>[,tecbname\|,(r)][,PREC] |

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 (bit 0 of the TECB is set to 1) and the task is posted ready to run if it is already waiting.

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.

  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

| Name | Operation | Operand |
|---|---|---|
| [name] | SETL | {filename\|(r1)}<br>,{id-name\|(r2)\|KEY\|BOF\|GKEY} |

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 25 on page 239.

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 | Identifier | 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 25. Field Supplied for SETL Processing by Record ID

## SETPFA (Set Link to Page-Fault Appendage) Macro

| Name | Operation | Operand |
|---|---|---|
| [name] | SETPFA | {entryaddress\|(0)} |

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 component program such as ACF/VTAM.

If your supervisor was generated with VM=YES (in the SUPVR generation macro), the macro results in a null operation.

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|(0)
: 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 SETPFA's for that task. Only one task per partition is allowed to have a page-fault appendage.

  The entry address may be specified as a symbol or in register notation.

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: |

| Bytes | Contents |
|---|---|
| 0 | Reserved. |
| 1-2 | Leftmost 16 bits of the 24-bit address of the page that is to be handled. |
| 3 | Space identifier |

All other 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.

Page Fault Queue: Your routine must maintain a queue of four-byte entries, one entry per private subtask controlled by your VSE task. Figure 26 on page 243 indicates how your appendage routine should manage the internal queue of page faults. Review Figure 26 after having read the remaining paragraphs about using the SETPFA macro.

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 the page-fault is not related to your program, 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 the page fault relates to your program.

2. Store the contents of register 13 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 | Address in Register 13 From Superv. | Return to Superv. by | Internal Page Fault Queue | Address in Register 13 To Superv. |
|---|---|---|---|---|
| Occurrence R1 | PF address R1 | BR 7 | PF address R1 | irrelevant |
| Occurrence R2 | PF address R2 | BR 7 | PF address R1<br>PF address R2 | irrelevant |
| Occurrence R3 | PF address R3 | BR 7 | PF address R1<br>PF address R2<br>PF address R3 | irrelevant |
| *Occurrence, any | any PF address | B 4(7) | PF address R1<br>PF address R2<br>PF address R3 | unchanged |
| Occurrence R4 | PF address R4 | BR 7 | PF address R1<br>PF address R2<br>PF address R3<br>PF address R4 | irrelevant |
| Completion R1 | PF address R1 | BR 7 | PF address R2<br>PF address R3<br>PF address R4 | PF address R2 |
| Occurrence R5 | PF address R5 | BR 7 | PF address R2<br>PF address R3<br>PF address R4<br>PF address R5 | irrelevant |
| Completion R2 | PF address R2 | BR 7 | PF address R3<br>PF address R4<br>PF address R5 | PF address R3 |

* Page fault not related to the partition.

PF = Page fault
Rn = Page fault request n (1, 2, and so on) of your partition

Figure 26. Internal Page-Fault Queue and Communication with the System

## SETT (Set Task Timer) Macro

| Name | Operation | Operand |
|---|---|---|
| [name] | SETT | {timervalue\|(1)} |

The macro sets the task timer. You can use the macro only if your supervisor was generated with TTIME=partition-ID specified in the FOPT generation macro.

A SETT macro can be issued only by the main task of the partition owning the task timer. If it is issued by a program running in a partition not owning the task timer, this program is canceled with an illegal SVC message.

SETT must not be used within an abnormal termination exit routine.

When a program is restarted from a checkpoint, a timer interval set by a SETT macro is not restarted.

timervalue|(1)
: Specifies the timer value in milliseconds. The largest value you can specify is 21474836. A register can be specified, and if it is, that register must contain the number of milliseconds in binary.

  The specified time interval is decremented only while the task is executing. When the specified time interval has elapsed, the task timer routine supplied in the STXIT TT macro receives control.

  If the address of a routine is not supplied to the supervisor by the time of the interruption, the interrupt is ignored.

## STXIT (Set Exit) Macro

The format of this macro is either of the following:

To establish linkage:

| Name | Operation | Operand |
|---|---|---|
| [name] | STXIT | {AB\|IT\|OC\|PC\|TT}<br>,{rtnaddr\|(0)},{savearea\|(1)}<br>[,OPTION={DUMP\|NODUMP\|EARLY}]<br>[,MFG={area\|(S,area)\|(r)}] |

To terminate linkage:

| Name | Operation | Operand |
|---|---|---|
| [name] | STXIT | {AB\|IT\|OC\|PC\|TT} |

The macro establishes or terminates linkage between the supervisor and an exit routine of your program for handling the specified condition. Linkage must be established before an interrupt occurs. Use the EXIT macro to return from these routines.

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 issue 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. Detailed information on the save area and interrupt status is given in the IBM publication Service Aids.

Upon entry to an exit routine, the supervisor sets up the following registers:

- For STXIT, register 0 contains, in its low-order byte, the cancel code passed by VSE/Advanced Functions. For a list of these codes, see the IBM publication VSE/System Package Messages and Codes.
- Register 1 points to the user-supplied save area, which contains the interrupt status information and the contents of registers 0 through 15.
- Register 15 points to the exit routine entry.

All other registers have to be reloaded 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. On entry to 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. 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 if RETURN=YES in the ATTACH macro. 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.

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. This assumes the subtask will use the attaching task's abnormal termination routine. However, the subtask may override this 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. However, the interruption exit is not taken if:

- The program issuing the STXIT macro instruction is a VTAM application program, and
- ACF/VTAM, a licensed IBM telecommunication program, is processing a request on behalf of this application program.

The exit will be taken when ACF/VTAM has completed the program's request.

An interval timer interruption is ignored if no exit linkage has been established.

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).

OC An operator communication interruption routine is entered when the operator presses the request key on the console and types the MSG command. In case of multitasking, only the main task can process this condition.

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. The specified exit is not taken if the program check occurs while ACF/VTAM is processing a request issued by the program.

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.

TT Linkage to a task timer interruption routine can be established only if TTIME=partition-ID was specified in the FOPT macro for supervisor assembly.

A task timer interruption routine is entered when the time interval specified in the SETT macro has elapsed. The STXIT (and EXIT) TT macro can be issued only by the main task of the partition owning the task timer. If it is issued by a program running in a partition not owning the task timer, the program is canceled and an error message indicating illegal SVC is printed.

A task timer interruption is ignored if no exit linkage has been established. A task timer interrupt is ignored if it occurs while a task timer exit routine is still processing. This can happen only if a short time interval was issued in your exit routine.

rtnaddr
: 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.

savearea
: The address of a 72-byte area in which the supervisor stores the old interrupt status information and general registers 0 through 15, in that order. Your program must have a separate save area for each routine that is included.

OPTION={DUMP|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)
  : When 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=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. |

OPTION=EARLY can be set only in a subtask or in the main task and cannot be transferred via the ATTACH (ABSAVE=) macro.

MFG={area|(S,area)|(r)}
: This operand is required if the program that issues the STXIT AB macro with the operands OPTION=NODUMP and EARLY is to be reenterable. The operand specifies the address of a 64-byte dynamic storage area which the system needs during execution of the macro. Registers 0 and 1 may not be used for register notation.

Figure 27 on page 250 shows what happens when one of the five exit conditions occurs while an STXIT routine is being processed within a particular partition.

**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.

| Routine being Processed | Interrupt Condition | | | | |
|---|---|---|---|---|---|
| | AB | IT | OC | PC | TT |
| AB | C | D | I | C | D |
| IT | S | E | H | H | H |
| OC | S | H | I | H | H |
| PC | S | H | H | T | H |
| TT | S | H | H | H | I |

C = The job is canceled immediately. The AB routine does not receive control again.

D = The interrupt is delayed. The IT or TT exit routine receives control after the EXIT AB macro is issued. If no EXIT AB is issued, the interrupt is ignored.

E = Handling of the new timer interrupt is delayed until the processing of the EXIT IT for the original interrupt is complete.

H = The condition is honored. When the newly called routine has finished processing, control returns to the interrupted routine.

I = The condition is ignored.

S = Execution of the routine is suspended. Control is transferred to the AB routine.

T = The job abnormally terminates. If an AB routine exists, that routine receives control; else, a system abnormal end occurs.

Figure 27. Effect of an AB, IT, OC, PC, or TT Interrupt During STXIT Routine Execution

## SUBSID (Subsystem Information Display) Macro

| Name | Operation | Operand |
|---|---|---|
| [name] | SUBSID | INQUIRY<br>,NAME={name1\|(S,name1)\|(r1)}<br>,AREA={name2\|(S,name2)\|(r2)}<br>,LEN={length\|(r3)}<br>[,LVLTEST={NO\|YES}]<br>[,MFG={name4\|(r4)}] |

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 prior-release 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).

## TECB (Timer Event Control Block) Macro

| Name | Operation | Operand |
|---|---|---|
| [name] | TECB | |

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 | 0 If 0: the time specified in SETIME has not elapsed.<br>If 1: the time specified in SETIME has elapsed.<br>1-7 Reserved. |
| 3 | Reserved. |

## TESTT (Test Task Timer) Macro

| Name | Operation | Operand |
|---|---|---|
| [name] | TESTT | [CANCEL] |

The macro can be used only if TTIME=partition-id was specified in the FOPT generation macro for supervisor assembly.

The macro is used to test the amount of time that has elapsed from a task timer interval set by a corresponding SETT. The TESTT macro returns the time remaining in the interval, expressed in hundredths of milliseconds in binary, in register 0.

The TESTT macro can be issued only by the main task of the partition owning the task timer. If it is issued by a program running in a partition not owning the task timer, the program is canceled and an error message indicating illegal SVC is printed.

CANCEL
If this is specified, the remaining time of the interval is canceled, and the task timer exit routine is not entered.

If the macro is issued without an operand, the macro must not contain a comment unless the comment begins with a comma.

## TPIN (Telecommunication Priority In) Macro

| Name | Operation | Operand |
|---|---|---|
| [name] | TPIN | |

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.

If your supervisor was generated with VM=YES (in the SUPVR generation macro), the macro results in a null operation.

## TPOUT (Telecommunication Priority Out) Macro

| Name | Operation | Operand |
|---|---|---|
| [name] | TPOUT | |

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.

If your supervisor was generated with VM=YES (in the SUPVR generation macro), the macro results in a null operation.

## TRUNC (Truncate Block) Macro

| Name | Operation | Operand |
|---|---|---|
| [name] | TRUNC | {filename\|(1)} |

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

| Name | Operation | Operand |
|---|---|---|
| [name] | TTIMER | [CANCEL] |

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

| Name | Operation | Operand |
|---|---|---|
| [name] | UNLOCK | [{name\|(S,name)\|(r)}\|ALL] |

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 macro). To use the UNLOCK macro for this purpose, you must issue the MODDTL macro with CHANGE=ON.

{name|(S,name)|(r)}
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

**Note:** UNLOCK ALL does not provide a return code, and register 15 remains unchanged.

0 Successful request; the resource has been unlocked.

4 The resource is not locked for the unlocking task.

8 DTL format error.

## VIRTAD (Virtual Address Return) Macro

| Name | Operation | Operand |
|---|---|---|
| [name] | VIRTAD | {address\|(1)} |

In 370 mode, 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.

In ECPS:VSE mode, only a virtual address can be specified as input. The macro returns that same address.

Register 0 returns the virtual address only if:

- For 370 mode, the specified real address points to a page frame that contains a PFIXed page,
- For ECPS:VSE mode, the specified 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.

**Note:** The pages of a program running in real mode are considered to be fixed.

## WAIT (Wait for Event) Macro

| Name | Operation | Operand |
|---|---|---|
| [name] | WAIT | {blockname\|(1)} |

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.

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.

For a discussion of the use of ECBs, see the IBM publication Application Programming: 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 XECB support when tasks belong to different, private partitions (but in the same address space).

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

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 WAIT 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.

**Note:** When a wait 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.

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 block set up for the I/O device and used in the EXCP macro for the device.

## WAITF (Wait for Completion of I/O) Macro

| Name | Operation | Operand |
|---|---|---|
| [name] | WAITF | {filename1\|(r1)}<br>[,filename2\|,(r2)],... |

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.

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

| Name | Operation | Operand |
|---|---|---|
| [name] | WAITM | {ecb1,ecb2,...\|listname\|(1)} |

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 macro for a description of the types of event control blocks and the restrictions on their use.

When control returns to a waiting task, register 1 points to the posted event control block (byte 2, bit 0 set to 1).

Note that a MICR CCB gets posted only when the device stops, not when a record is read. Furthermore, telecommunication ECBs must not be waited for, because their format does not satisfy a WAIT or a WAITM (that is, bit 0 of byte 2 would not be posted).

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 full-word addresses that point to the ECBs to be waited upon. The first byte following the last address in the list must be nonzero to indicate the end of the list.

## WRITE (Write a Record) Macro

| Name | Operation | Operand |
|---|---|---|
| [name] | WRITE | {filename\|(1)}, |
| | | {{SQ\|UPDATE}, |
| | | {area\|(0)}[,length\|,(r)]\| |
| | | AFTER[,EOF]\| |
| | | ID\| |
| | | KEY\| |
| | | NEWKEY\| |
| | | RZERO} |

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 and 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. Is used only for 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.

## XECBTAB (Cross-Partition Event Control Block Table) Macro

| Name | Operation | Operand |
|---|---|---|
| [name] | XECBTAB | TYPE={DEFINE\|DELETE\|<br>CHECK\|RESET\|DELETALL}<br>,XECB=xecbname<br>[,XECBADR={xecbfield\|<br>(S,xecbfield)\|(r1)}]<br>[,ACCESS={XPOST\|XWAIT}]<br>[,MFG={area\|(S,area)\|(r2)}] |

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 partitions in the same address space.

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 28 on page 263.

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 28 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.

| TYPE= | X'00' | X'04' | X'08' |
|---|---|---|---|
| CHECK | The named XECB was found in the table (see also Note 1 on page 264). | The named XECB was not found (see also Note 2 on page 264). | The named XECB was found in the table (see also Note 3 on page 264). |
| DEFINE | The named XECB is stored in the table (see also Note 1 on page 264). | The named XECB is already in the table (see also Note 2 on page 264). | The named XECB is full (see also Note 2 on page 264). |
| DELETE | The named XECB is removed from the table (see also Note 2 on page 264). | The named XECB was not found (see also Note 2 on page 264). | The issuing program did not define the XECB (see also Note 2 on page 264). |

Figure 28 (Part 1 of 2). XECBTAB Feedback Information

| TYPE= | X'00' | X'04' | X'08' |
|---|---|---|---|
| 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). |

Figure 28 (Part 2 of 2). XECBTAB Feedback Information

**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 partition can continue to use an XECB to communicate with VSE/POWER, a licensed IBM program that runs in a shared partition.

## XPOST (Cross-Partition Post) Macro

| Name | Operation | Operand |
|---|---|---|
| [name] | XPOST | XECB={xecbname\|(1)}<br>POINTRG=(14) |

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 main-line 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 main-line 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 29 on page 268, 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:

X'00' Successful completion. The named XECB has been posted.

X'04' The named XECB has no associated table entry in the XECB table.

X'0C' The named XECB is not within the current address space.

X'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.

X'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

| Name | Operation | Operand |
|---|---|---|
| [name] | XWAIT | XECB={xecbname\|(1)}<br>POINTRG=(14) |

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 29 on page 268 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 29. Coding Example Showing the Use of XECBTAB with TYPE=CHECK and XWAIT

# 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 other than IBM 2560, 5424, and 5425 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 1" 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 1" 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 (see "Note 2" below) |
| W | Select stacker 2 (see 'Note 3" below |
| X | Select stacker 3 (see "Note 4" below) |
| Y | Select stacker 4 (see "Note 4" below) |
| X | Select stacker 5 (see "Note 5" below) |

**Notes:**

1. 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.

2. For a DTFDI defined file on an IBM 2560, 5424, or 5425:

   Primary hopper, select stacker 1
   Secondary hopper on 2560, select stacker 5
   Secondary hopper on 5424 or 5425, select stacker 4

3. For a DTFCD defined file on an IBM 2560, 5424, or 5425:

   Primary hopper, select stacker 2
   Secondary hopper, select stacker 3

4. Applies only to a DTFCD defined file on an IBM 2560, 5424, or 5425.

5. Applies only to a DTFCD defined file on an IBM 2560.

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 an IBM 1442 (see "Note 1" below): | | |
| 81 | 12-0-1 | Select into stacker 1 |
| C1 | 12-1 | Select into stacker 2 |
| Stacker Selection on Another Card I/O Device: | | |
| 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 2" 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 |
| 53 | 12-11-3-9 | Primary hopper: select into stacker 5 (see "Note 3" below). |
| 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 |
| D3 | 11-3 | Secondary hopper: select into stacker 5 (see "Note 3" below) |

**Notes:**

1. Use the stacker-selection codes for the IBM 2520 if you defined your file with DTFDI.
2. The code cannot be used if you defined your file with DTFDI.
3. Applies only to a file on an IBM 2560.

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 30 on page 276 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 31 on page 277 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.

| $b_7$ → | | | | | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $b_6$ → | | | | | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| $b_5$ → | | | | | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| Bits ③ $b_4$ | $b_3$ | $b_2$ | $b_1$ | Column → / Row ↓ | 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: Pound 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 Text (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 Characters

| | | | |
|---|---|---|---|
| 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 (This graphic is stylized to distinguish it from Logical OR) |
| - | Hyphen (Minus) | } | Closing Brace |
| . | Period (Decimal Point) | ~ | Tilde |
| / | Slant | | |
| : | Colon | | |
| ; | Semicolon | | |

Figure 30. ASCII Character Set

| ASCII | | | | EBCDIC | | | |
|---|---|---|---|---|---|---|---|
| Character | Col | Row | Bit Pattern | Col (in Hex) | Row (in Hex) | Bit Pattern | Comments |
| 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 | |
| ! 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 |
| . | 2 | 14 | 0010 1110 | 4 | B | 0100 1011 | |
| / | 2 | 15 | 0010 1111 | 6 | 1 | 0110 0001 | |
| 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 | |

Figure 31 (Part 1 of 2). ASCII to EBCDIC Correspondence

| ASCII | | | | | EBCDIC | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Character | Col | Row | Bit Pattern | | Col (in Hex) | Row | Bit Pattern | | Comments |
| @ | 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 | |
| \ | 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 | Underscore |
| ` | 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 | |
| \| ① | 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).

Figure 31 (Part 2 of 2). ASCII to EBCDIC Correspondence

# APPENDIX C. STANDARD AND NON-STANDARD LABELS

This appendix assumes that you are familiar with the concepts of label processing as described in the IBM publication System Management Guide. It summarizes the coding requirements in a user-written routine for the processing of user-standard labels or non-standard labels. It 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 non-standard 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 non-standard 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:

  0 = 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 non-standard 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:
  - Non-standard 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 0. This indicates to your routine that non-standard 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 tape mark, 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 non-standard 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 32 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 32.

Additional volume labels, if any, are ignored by VSE.

| 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. |

Figure 32 (Part 1 of 2). Disk Volume Label (VOL1)

| Displ. | Length | Content |
|---|---|---|
| 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. |

Figure 32 (Part 2 of 2). Disk Volume Label (VOL1)

## 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 33 on page 284 - The first IBM standard disk file label (format-1).

- Figure 34 on page 286 - The IBM standard disk file continuation label (format-3).
- Figure 35 on page 287 - 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 35.

| 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: yyddd. 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 OS/VS. |
| 3D | 01 | Reserved. |
| 3E | 0D | System code: IBMDOSVS. Written by IOCS. |
| 4B | 03 | Date of last access: yyddd; not used by VSE. |
| 4E | 02 | Reserved |
| 50 | 02 | Number of blocks per CI for FBA, blanks for CKD. |

Figure 33 (Part 1 of 3). IBM-Standard Disk File Label (Format-1)

| Displ. | Length | Content |
|---|---|---|
| 52 | 02 | File type (from DLBL; checked against the type of DTF):<br><br>X'0008' = VSAM<br>X'2000' = DAM<br>X'4000' = SAM (default)<br>X'4100' = A library file whose file name begins with the characters IJSYS<br>X'8000' = ISAM |
| 54 | 01 | Used by OS/VS. IOCS writes 0. |
| 55 | 01 | Flags for optional areas used by ISAM files (from DTF and EXTENT):<br><br>Bit 2 = 1: Master index.<br>Bit 3 = 1: Independent overflow area.<br>Bit 4 = 1: Cylinder overflow area. |
| 56 | 02 | Byte length of ISAM blocks. From DTF. |
| 58 | 02 | Record length of ISAM files. From DTF. |
| 5A | 01 | Key length of ISAM blocks. From DTF. |
| 5B | 02 | Key field location in ISAM block. From DTF. |
| 5D | 01 | Flags:<br><br>Bit 0 = 1: Last volume (SAM only).<br>Bit 3 = 1: File security. From DLBL. |
| 5E | 01 | Original space request was:<br><br>Bit 1 = 1: In blocks.<br>Bit 4 = 1: For continuous extent.<br>Bit 5 = 1: For maximum continuous extent.<br>Bit 6 = 1: Not less than the specified minimum. |
| 5F | 03 | Used by OS/VS. IOCS writes blanks. |
| 62 | 05 | Used by OS/VS. IOCS writes zeros. |
| 67 | 02 | Start of next record to end-of-data distance. |

Figure 33 (Part 2 of 3). IBM-Standard Disk File Label (Format-1)

| Displ. | Length | Content |
|---|---|---|
| 69 | 01 | Type of extent (from EXTENT):<br><br>X'01' = Prime data area or data space extent the default).<br>X'02' = Independent overflow area.<br>X'04' = Master/cylinder index area extent.<br>X'40' = Extent for user-standard labels.<br>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 33 (Part 3 of 3). IBM-Standard Disk File Label (Format-1)

| Displ. | Length | Content |
|---|---|---|
| 00 | 04 | Key code for continuation label (03030303). Written by IOCS. |
| 04 | 01 | Type of extent, from EXTENT:<br><br>X'01' = Data extent (default).<br>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. |

Figure 34 (Part 1 of 2). IBM-Standard Disk File Continuation Label (Format-3)

| Displ. | Length | Content |
|---|---|---|
| 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 34 (Part 2 of 2). IBM-Standard Disk File Continuation Label (Format-3)

| 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 OS/VS. |
| 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:<br>Bit 0 = 1: Always set to 1.<br>Bit 3 = 1: Volume reserved for emulators.<br>Bit 5 = 1: VTOC is being updated by VSAM. |
| 3B | 01 | Extent count. Always 1 (The VTOC is one extent). |
| 3C | 02 | Reserved. |
| 3E | 0E | CKD device constants (for FBA: zeros): |

Figure 35 (Part 1 of 3). Disk VTOC Label (Format-4)

| Displ. | Length | Content |
|---|---|---|
| 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. |
| 47 | 01 | Flag byte:<br><br>Bit 4 = 1: An I or L value takes two bytes.<br>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:<br><br>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 OS/VS. |
| 60 | 04 | Number of first block of FBA catalog recovery area. For CKD: zeros. |
| 64 | 05 | Reserved. |

Figure 35 (Part 2 of 3). Disk VTOC Label (Format-4)

| Displ. | Length | Content |
|---|---|---|
| 69 | 01 | Extent type: 01 for VTOC extent. |
| 6A | 01 | Extent sequence number:<br>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. |

Figure 35 (Part 3 of 3). Disk VTOC Label (Format-4)

## 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 36 on page 290 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 37 on page 290.

| Displ. | Length | Content |
|---|---|---|
| 00 | 04 | Key area:<br><br>UHLn for a header label.<br>UTLn for a trailer label.<br><br>where n is the label-sequence number:<br><br>A number from 1 to 8 for header labels.<br>A number from 0 to 7 for trailer labels. |
| 04 | 04 | Data area:<br><br>UHLn for a header label.<br>UTLn for a trailer label.<br><br>where n = The label-sequence number: a number from 1 to 8 for all user labels. |
| 08 | 4C | The user's label information |

Figure 36. User-Standard Disk-File Label (Header and Trailer)

| 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 |

Figure 37 (Part 1 of 2). User-Standard Disk-File Label (Five UHLs and Four UTLs are Specified)

| Key Area | Data Area |
|---|---|
| UTL3 | UTL4 + 76 bytes of user label data |
| UTL4 | |
| User data | |

Figure 37 (Part 2 of 2). User-Standard Disk-File Label (Five UHLs and Four UTLs are Specified)

If only header labels are specified, one UTL0 label without data is written by the system. An example in Figure 38 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.

| 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 | |

Figure 38. User-Standard Disk-File Label (Three UHLs are Specified)

## 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 39 shows the format of a 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. |

Figure 39. Diskette Volume Label

## 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 40 shows the format of a 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. |

Figure 40 (Part 1 of 3). Diskette File Label

| Displ. | Length | Content |
|---|---|---|
| 16 | 05 | Record length. From IOCS. |
| 1B | 01 | Reserved. |
| 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:<br>blank = Sector length is 128 bytes, unblocked, non-spanned, sequential.<br>non-blank = Job ends on input. |
| 2C | 01 | Volume byte:<br>blank = File complete on this volume.<br>C = File continued on next volume.<br>L = File ends on this volume. |
| 2D | 02 | Volume sequence number. |
| 2F | 06 | Creation date: yymmd. |
| 35 | 0D | Reserved. |
| 42 | 06 | Expiration date: default = seven days after output. |
| 48 | 01 | Verify byte: V or blank. |
| 49 | 01 | Reserved. |
| 4A | 05 | End-of-data address. |

Figure 40 (Part 2 of 3). Diskette File Label

| Displ. | Length | Content |
|---|---|---|
| 4F | 01 | Reserved. |

Figure 40 (Part 3 of 3). Diskette File Label

## 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 41 and Figure 42 show volume labels for EBCDIC and ASCII tapes.

| 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 41. Tape Volume Label for EBCDIC 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. |

Figure 42 (Part 1 of 2). Tape Volume Label for ASCII Code

| Displ. | Length | Content |
|---|---|---|
| 33 | 1C | Reserved. |
| 4F | 01 | Standard byte:<br>1 = File has ANSI standard.<br>blank = File does not have ANSI standard. |

Figure 42 (Part 2 of 2). Tape Volume Label for ASCII Code

## 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 - At the beginning of a file.
  - HDR2 - At the beginning of a file (see "Note" below).
- Trailer labels:
  - EOF1 - At the end of a file.
  - EOF2 - At the end of a file (see "Note" below).
  - EOV1 - At the end of a volume and not the end of a file.
  - EOV2 - At the end of a volume and not the end of a file (see "Note" below).

**Note:** The label is used only with ASCII files.

Additional labels (HDR3 through HDR8, if present) are ignored by VSE.

Figure 43 through Figure 45 on page 297 show IBM-standard file labels for tapes.

| 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 | Sub-version 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 | 07 | Reserved. |

Figure 43. 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 within volume(s). |
| 23 | 04 | Version number of the file. |

Figure 44 (Part 1 of 2). First IBM-Standard Tape File Label for ASCII Code

| Displ. | Length | Content |
|---|---|---|
| 27 | 02 | Sub-version 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 44 (Part 2 of 2). First IBM-Standard Tape File Label for ASCII Code

| Displ. | Length | Content |
|---|---|---|
| 00 | 04 | Label ID: HDR1, EOF1, or EOV1. |
| 03 | 01 | Label sequence number: 2. |
| 04 | 01 | Record format:<br>C'F' for fixed.<br>C'V' for variable.<br>C'U' for undefined length. |
| 05 | 05 | Block length. |
| 0A | 05 | Record length. |
| 0F | 23 | Reserved. |
| 32 | 02 | Buffer-offset length. |
| 34 | 1C | Reserved. |

Figure 45. Second IBM-Standard Tape File Label for ASCII Code

## 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

Non-standard labels are supported only on EBCDIC code tape labels. Non-standard 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 non-standard 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.

# INDEX


A

AB exit routine
  return from 149
abnormal end
  See AB exit routine
abnormal end exit 246
ABSAVE operand (ATTACH macro) 8
ACCESS operand (XECBTAB) 263
activate
  partition (TPOUT) 253
  task (ATTACH) 6
ADDAREA operand (DTFMR) 100
address delimiter 50
ADDRESS operand
  DTFMR macro 101
  MRMOD macro 200
address specification
  dynamically requested storage 169
  FREEVIS macro 159
  GENDTL 161
  I/O request block (GENIORB) 164
  job communication area (JOBCOM) 181
  local directory list 165
  lock file, for modification 198
  lock file, generate dynamically 161
  MODDTL 198
  PDUMP macro 208
  storage, dynamically requested 169
  virtual storage requested 169
address translation
  real address return (REALAD) 227
  virtual address return 256
AFTER operand (DTFDA) 66
argument list (CP CLOSE) 41
ASA option control characters 269
ASCII character code 275
ASCII character set 276
ASCII file
  block prefix 104
  file (DTFMT) 103
  file (DTFPH) 119
  file label for 296
  volume label for 294
ASCII to EBCDIC correspondence 277
ASPL operand 5
assign
  device (ASSIGN) 5
  parameter list (ASPL) 5
associated file
  cardI/O definition (of DTFCD) 54
  print definition (of DTFPR) 124
  print module (PRMOD) 215
  print output 124
associated file (ASOCFLE) operand
  card I/O 55
  DTFCD macro 55
  DTFPF macro 124
  print output 124
attach (a task) 6

B

backspace tape 37
backward read (from tape) 108
batch and serial number (BCHSER) operand (DFR) 44
batch number (BCH) operand (DFR) 44
BINARY operand (GETIME) 168
block length
  See also block size (BLKSIZE) operand
  card I/O 56
  combined file output 60
  console I/O 63
  direct access file 67
  disk sequential I/O 132
  document-read (3886) file 78
  indexed-sequential file 97
  print output 125
  tape file 103
  1287/1288 input 112
block size (BLKSIZE) operand
  card I/O 56
  console I/O 63
  direct access file 67
  disk sequential I/O 132

DTFCD macro 56
DTFCN macro 63
DTFDA macro 67
DTFDR macro 78
DTFMT macro 103
DTFOR macro 112
DTFPR macro 125
DTFSD macro 132
print output 125
tape I/O 103
1287/1288 input 112
3886 input 78
blocking factor (BLKFAC) operand
DTFOR macro 111
ORMOD macro 204
1287/1288 input 111
1287/1288 input module 204
buffer
additional, MICR file 100
indexed sequential add 96
MICR document 27
MICR/OCR input (DTFMR) 101
print, clear the 38
status (MICR) 27
synchronize, tape 37
buffer offset (BUFOFF) operand (DTFMT) 104
BUFFERS operand
DTFMR macro 101
MRMOD macro 201

C

call (a program) 9
cancel program
DUMP macro 144
JDUMP macro 179
STXIT option 248
task 10
task-timer interval 252
card error (CRDERR) operand
card I/O 56
CDMOD macro 21
DTFCD macro 56
card I/O
define I/O module for 21
device control 39
file, define a (DTFCD) 54
I/O 62
carriage spacing, printer 37
CCB
See command control block (CCB)
CCW address (CCWADDR) operand (DTFPH) 120
CCW operand
GENIORB macro 164
IORB macro 173
CDLOAD macro 19
CDMOD macro 21
CHANGE operand (MODDTL) 198
channel program
command control block for 11
request to execute (EXCP) 148
CHAP macro 25
character
control
See control characters
table reference 77, 130, 218
character delimiter 50
character set options (3886 file) 45
character-set (CHRSET) operand (DFR) 45
characters, control 269
CHECK macro 26
checkpoint
end address 33
file on disk 119
on tape, interspersed 104
request macro (CHKPT) 32
restart address 32
CISIZE specification
See control-interval size
CKPTREC operand (DTFMT) 104
CLOCK operand (GETIME) 168
close a file (CLOSE, CLOSER) macro 34
CMDCHN operand (DTFDU) 82
code, ASCII character 275
codes, control character 269
combined file
card I/O (DTFCD) 54
console (DTFCN) 63
read from (GET) 166
write to (PUT) 221
command chaining, diskette I/O 82
command control block (CCB)
address of in EXCP 148
conditions, testable 14
define a (CCB) 11
fromat of 12
transmission information 14
command list
CCB macro 11
in IORB (CCW operand) 164, 173

comment specification 2
communication
  See also cross-partition communication
  activate partition (TPOUT) 253
  deactivate partition (TPIN) 253
  immediate response request 253
  job (step) to job (step) 180
  operator interrupt 247
  region access (COMRG) macro 41
  region, move data to 201
composed page data stream (CPDS) records 125
compressed fix list 174
console file
  combined, write to (PUT) 223
  define a (DTFCN) 63
  output, write to (PUT) 221
  read from (GET) 166
control
  See also return control
  See also virtual storage control
  a resource (LOCK) 190
  card device 21
  character (CDMOD) 21
  characters, ASA option 269
  characters, EBCDIC option 271
  code, I/O command 36
  device operation (CNTRL) macro 35
  overflow of print page 220
  page overflow 129, 217
  pocket lights 185
  print output 126, 215
  printer data check 131
  protected resource 140
  protected resource, dynamic setting 161
  protected resource, modify 199
  return of 232
  spanned-record file 37
  unit, secondary (MICR input) 102
control block
  See control information
control character (CTLCHR) operand
  card I/O 56
  CDMOD macro 21
  DTFCD macro 56
  DTFPR macro 127
  print output 127
  print output module 216
  PRMOD macro 216
control characters
  define for print output 127, 216
control information
  device (unit), DSECT for 171
  DTFPH table 123
  look-up (EXTRACT macro) request 150
  partition boundary DSECT 194, 195
  partition boundary format 195, 196
  XTNTXIT information 123
CONTROL operand
  card I/O 56
  CDMOD macro 21
  direct access file 67
  DTFCD macro 56
  DTFDA macro 67
  DTFOR macro 112
  DTFPR macro 126
  DTL macro 140
  GENDTL macro 161
  MODDTL macro 199
  ORMOD macro 204
  print output 126
  print output module 215
  PRMOD macro 215
  resource lock/unlock 140
  resource, lock/unlock, dynamic setting 161
  resource, lock/unlock, modify 199
  1287/1288 input 112
  1287/1288 input module 204
control-address specification (CHECK) 26
control-interval size
  device independent I/O 73
  DTFDI macro 73
  DTFPH macro 120
  DTFSD macro 133
  physical IOCS 120, 121
  sequential file on FBA 133
conventions, notational 2
CORDATA operand (ISMOD) 175
CORINDX operand (ISMOD) 175
correction exit (COREXIT) operand
  DTFDR macro 78
  DTFOR macro 112
  1287/1288 input 112
  3886 input 78
CPCLOSE macro 41
CPDS records 125
cross-partition communication
  event control block (XECBTAB) 261
  example for use 268
  post event (XPOST) 264
  wait on event (XWAIT) 266

cylinder overflow (CYLOFL) operand (DTFIS) 88

D

data block
See block length
data check control 131
data-check control, printer 38
DDKR operand (SECTVAL) 234
DE operand
FETCH macro 156
GENL macro 166
LOAD macro 188
deactivate
partition (TPIN) 253
task 43
define
card I/O file (DTFCD) 54
console I/O file (DTFCN) 63
device-independent file (DTFDI) 73
direct access file (DTFDA) 66
diskette-unit file (DTFDU) 82
file for physical IOCS (DTFPH) macro 119
indexed-sequential file (DTFIS) 88
MICR/OCR file (DTFMR) 100
OCR file (DTFOR) 111
printer file (DTFPR) macro 124
sequential disk file (DTFSD) macro 132
tape file (DTFMT) 103
VSAM managed sequential 132
3886 document-read file (DTFDR) 78
define a file
See file definition
DELETFL operand (DTFSD) 134
dequeue resource (DEQ) macro 42
detach a task (DETACH) macro 43
device
control (CNTRL) 35
device address (DEVADDR) operand
card I/O 57
console I/O 63
device independent I/O 73
direct access file 67
disk sequential I/O 134
diskette I/O 82
DTFCD macro 57
DTFCN macro 63
DTFDA macro 67
DTFDI macro 73
DTFDR macro 79
DTFDU macro 82
DTFMR macro 101
DTFMT macro 105
DTFOR macro 115
DTFPH macro 120
DTFPR macro 127
DTFSD macro 134
MICR/OCR input 101
physical IOCS 120
print output 127
tape I/O 105
1287/1288 input 115
3886 input 79
device independence
define I/O module for 46
file for, define a (DTFDI) 73
DEVICE operand
card I/O 57
CDMOD macro 22
diskette I/O 83
DRMOD macro 52
DTFCD macro 57
DTFDR macro 80
DTFDU macro 83
DTFIS macro 89
DTFOR macro 115
DTFPH macro 120
DTFPR macro 127
GENIORB macro 164
I/O request block 164, 173
indexed sequential file 89
IORB macro 173
ORMOD macro 204
physical IOCS 120
print output 127
print output module 216
PRMOD macro 216
1287/1288 input 115
1287/1288 input module 204
DIMOD macro 46
direct access file
define a (DTFDA) 66
read from (READ) 226
trailer labels for 71
wait for end of I/O 258
write to (WRITE) 259
directory list, local
See local directory list
directory search only (TXT of LOAD) 189

disengage document feed (DISEN) macro 48
refid=doc.feed, stop of (DISEN) 48
disk area, free a (FREE) 158
disk device
device-independent file on (DTFDI) 73
direct access file on (DTFDA) 66
dynamic assign of 5
indexed-sequential file on (DTFIS) 88
physical IOCS file on (DTFPH) 119
restart address (CHKPT) 33
seek address operation 36
sequential file on (DTFSD) 132
disk file
device independent (DTFDI) 73
direct access (DTFDA) 66
force end of volume (FEOVD) 155
indexed sequential (DTFIS) 88
physical IOCS (DTFPH) 119
release block (RELSE) 231
sequential (DTFSD) 132
VSAM managed sequential 132
disk file label
continuation 286
first (only) 284
format-1 284
format-3 286
format-4 287
standard 283
user-standard 290
VTOC label 287
diskette label
file, standard 292
volume 291
diskette-I/O file
define I/O module for 142
file, define a (DTFDU) 82
display document field (DSPLY) macro 53
DLINT macro 49
document
buffer 27
control 40
field, display of (DSPLY) 53
read (3886) file (DTFDR) 78
read/sort file (DTFMR) 100, 111
DRMOD macro 52
DSECT operand (IORB) 173
DSKXTNT operand
direct access file 67
DTFDA macro 67
DTFIS macro 89
indexed sequential file 89
DTFCD macro 54
DTFCN macro 63
DTFDA macro 66
DTFDI macro 73
DTFDR macro 78
DTFDU macro 82
DTFIS macro 88
DTFMR macro 100
DTFMT macro 103
DTFOR macro 111
DTFPH macro 119
DTFPR macro 124, 215
DTFSD macro 132
DTL (define the lock) macro 140
DTL (lock control block)
See lock control block
dual address adapter
definition for 101
module definition for 200
DUMODFx macro 142
DUMP macro 144
dump request by
CANCEL macro 10
DUMP macro 144
JDUMP macro 179
PDUMP macro 208
DVCTYP operand (SECTVAL) 235
dynamic generation
directory list 165
I/O request block (IORB) 163
lock control block 161

E

EBCDIC option control characters 271
EBCDIC to ASCII correspondence 277
ECB operand
ATTACH macro 8
GENIORB macro 164
IORB macro 173
PAGEIN macro 207
edit character (EDCHAR) operand (DFR) 45
EDITn operand (DLINT) 51
eject document 40
end address (CHKPT) 33
end file load mode (ENDFL) macro 145
end of volume

disk, force end of (FEOVD) 155
force, DTFSD 136
records, handling by DTFDA 68
SYSLST/SYSPCH on tape (SEOV) 235
tape, force end of (FEOV) 154
end-of-extent exit routine
physical I/O 121
sequential I/O 134
end-of-file exit routine
card I/O 57
device independent I/O 73
diskette I/O 83
tape I/O 105
1287/1288 input 115
3886 input 80
end-of-job (EOJ) macro 146
end-set-limit (ESETL) 147
enqueue resource (ENQ) macro 145
entry-point specification
CALL macro 9
EOFADDR operand
card I/O 57
device independent I/O 73
disk sequential I/O 134
diskette I/O 83
DTFCD macro 57
DTFDI macro 73
DTFDR macro 80
DTFDU macro 83
DTFMT macro 105
DTFOR macro 115
DTFSD macro 134
tape I/O 105
1287/1288 input 115
3886 input 80
EOXPTR operand
DTFPH macro 121
DTFSD macro 134
physical I/O 121
sequential I/O 134
erase gap 37
ERASE operand (DFR) 46
erase-character recognition 46
ERRBYTE operand (DTFDA) 67
ERREXT operand
direct access file 68
diskette I/O 83, 142
DTFDA macro 68
DTFDU macro 83
DTFIS macro 89
DTFMT macro 105
DUMODFx macro 142
indexed sequential file 89
indexed sequential I/O module 176
ISMOD macro 176
tape I/O 105
ERROPT operand
See error option
error exit routine
card I/O 58
device independent I/O 74
diskette I/O 83
IORB generation 165
print output 128
return from 147
sequential disk I/O 135
tape I/O 106
1287/1288 input 112
3886 input 78
error option
card I/O 57
device independent I/O 74
disk sequential I/O 135
diskette I/O 83
diskette I/O module 142
DTFCD macro 57
DTFDI macro 74
DTFDU macro 83
DTFMR macro 101
DTFMT macro 105
DTFPR macro 128
DTFSD macro 135
DUMODFx macro 142
MICR/OCR input 101
print output 128
print output module 216
PRMOD macro 216
resource-lock failure 191
tape I/O 105
error retry
card I/O (CRDERR of DTFCD) 56
error-return (ERET) macro 147
errtry.card I/O (CRDERR of CDMOD) 21
ESETL macro 147
event control block
See also ECB operand
format of 8
IORB controlled I/O 164, 173
post an (POST) 214
timer 252
execute-channel-program (EXCP) macro 148
exit indicator (EXITIND) operand (DTFDR) 80
exit routine
See also exit routine linkage

abnormal end 246
end of extent
See end-of-extent exit routine
end of file
See end-of-file exit routine
error
See error exit routine
interval timer interrupt 247
label processing
See labels, user
MICR/OCR stacker select 101
operator communication interrupt 247
page fault (SETPFA) 240
page overflow 221
program check interrupt 247
return control (EXIT) 149
return from 232
STXIT defined 245
task timer interrupt 247
wrong-length record
See wrong-length-record exit routine
exit routine linkage
abnormal end 246
interval timer interrupt 247
operator communication interrupt 247
program check interrupt 247
set up (STXIT) 245
task timer interrupt 247
EXTADDR operand (DTFMR) 101
extent checking, direct access file 72
extent checking, physical IOCS 123
extent information 72
extract control information
macro (EXTRACT) 150
partition boundary DSECT 194, 195
unit information DSECT (IJBPUB) 171


F


FAIL operand (LOCK) 191
FEED operand (DTFDU) 85
FEOVD operand
See force end of volume
FETCH (a phase) macro 155
FILABL operand (DTFMT) 107
file
See also file definition
ASCII, logical IOCS access (DTFMT) 103
ASCII, physical IOCS access (DTFPH) 119
associated (DTFCD) 54
associated, card I/O (DTFCD) 54
associated, card print (DTFPR) 124
card I/O (DTFCD) 54
close a 34
combined (DTFCD) 54
combined (DTFCN) 63
console (DTFCN) 63
console, write to and read (PUTR) 223
device independent (DTFDI) 73
direct access (DTFDA) 66
diskette I/O (DTFDU) 82
diskette I/O module 142
document read (1287/1288) (DTFOR) 111
document read/sort (DTFMR) 100
document-read (3886 - DTFDR) 78
indexed sequential I/O module 175
indexed sequential, load a (ENDFL) 145
indexed sequential, load mode (SETFL) 236
indexed-sequential (DTFIS) 88
MICR/OCR I/O module (MRMOD) 200
open a 203
physical IOCS (DTFPH) 119
printer (DTFCD) 124
read from (READ) 226
read from, sequential (GET) 166
sequential disk (DTFSD) 132
tape (DTFMT) 103
write to (WRITE) 259
write to, sequential (PUT) 221
3881 (DTFCD) 54
file definition
ASCII, logical IOCS access (DTFMT) 103
ASCII, physical IOCS access (DTFPH) 119
associated, card print (DTFPR) 124
associated, print module (PRMOD) 215
combined (DTFCN) 63
console (DTFCN) 63
device independent (DTFDI) 73
direct access (DTFDA) 66

diskette I/O (DTFDU) 82
document read (1287/1288) (DTFOR) 111
document read/sort (DTFMR) 100
document-read (3886 - DTFDR) 78
indexed-sequential (DTFIS) 88
physical IOCS (DTFPH) 119
printer (DTFCD) 124
sequential disk (DTFSD) 132
tape (DTFMT) 103
3881 file (DTFCD) 54
file label
disk
See disk file label
diskette
See diskette label
tape
See tape label
file position control
note record address (NOTE) 202
point behind noted record (POINTW) 213
point to noted record (POINTR) 212
point to start of file (POINTS) 213
file-security (FILESEC) operand (DTFDU) 85
file-type specification
See type of file (TYPEFLE) operand
filename specification
CHECK macro 26
CHKPT macro 34
CLOSE (CLOSER) macro 35, 203
CNTRL macro 36
DISEN macro 48
DSPLY macro 53
fix a page (PFIX) macro 209
FIXFLAG operand (IORB) 174
FIXLIST operand (IORB) 173
FLDn operand (DLINT) 50
FONT operand (DFR) 44
force end of volume
direct access file 68
disk (FEOVD) 155
disk sequential I/O 136
operand, DTFDA macro 68
operand, DTFSD macro 136
tape (FEOV) 154
force page out (FCEPGOUT) macro 152
format end (FREND) specification (DLINT) 52
format record (3886 file)
change of (SETDEV) 236
define a (DFR) 44
length of 80
line type of 49
phase name of 80
format record, change of (SETDEV) 236
forms control buffer
load message for 184
load request (LFCB) macro 183
FORMS operand (LFCB) 184
forward read (from tape) 108
forward space tape 37
free
disk area (FREE) 158
page (PFREE) 211
resource (DEQ) 42
virtual storage
See virtual storage release
virtual storage (FREEVIS) 159
FREND operand (DLINT) 52
FRNAME operand (DTFDR) 80
function operand
associated (print output) file (FUNC) 128, 217
card I/O (FUNC) 59
CDMOD macro (FUNC) 22
DTFCD macro (FUNC) 59
DTFIS macro (IOROUT) 95
DTFPR macro (FUNC) 128
indexed sequential file 95
indexed sequential I/O module 176
ISMOD macro 176
job-communication 181
PRMOD macro (FUNC) 217
UPDATE (DTFSD) 138

## G

gap, erase a 37
GENDTL macro 161
generate I/O request block (GENIORB) macro 163
generation, dynamic
See dynamic generation
GENL macro 165
GET macro 166
get time of day (GETIME) macro 167
get virtual storage
See virtual storage request
get virtual storage (GETVIS) macro 169

Greenwich mean time 168

H

HDRINFO operand
  DTFMT macro 107
  DTFPH macro 121
  physical IOCS 121
  tape I/O 107
header information
  print for physical IOCS 121
  print for tape I/O 107
HEADER operand
  DTFDR macro 81
  DTFOR macro 116
  1287/1288 input 116
  3886 input 81
high index (HINDEX) operand (DTFIS) 92
HOLD operand
  direct access file 68
  disk sequential I/O 136
  DTFDA macro 68
  DTFIS macro 92
  DTFSD macro 136
  indexed sequential file 92
  indexed sequential I/O module 176
  ISMOD macro 176
hopper empty (1287D, 1288) 116
horizontal copy control 38
HPRMTY operand (DTFOR) 116

I

I/O request block (IORB)
  address of in EXCP 148
  assembly of, in program 172
  define dynamically 163
I/O request block (IORB) macro 172
I/O-area size
  See block size (BLKSIZE) operand
I/O-area specification
  additional (DTFIS) 95
  card I/O 59
  CDMOD macro 22
  console file 64
  DIMOD macro 46
  direct access file 68
  disk sequential I/O 136
  diskette I/O 85
  DTFCD macro 59
  DTFDA macro 68
  DTFDI macro 75
  DTFDR macro 81
  DTFDU macro 85
  DTFMR macro 101
  DTFMT macro 107
  DTFOR macro 116
  DTFPR macro 129
  DTFSD macro 136
  indexed sequential I/O module 176
  ISMOD macro 176
  load file (DTFIS) 94
  MICR/OCR input 101
  ORMOD macro 204
  print output 129
  print output module 217
  PRMOD macro 217
  randomly process file (DTFIS) 94
  sequentially process file (DTFIS) 94
  tape I/O 107
  1287/1288 input 116
  1287/1288 input module 204
I/O-register specification
  card I/O 59
  disk sequential I/O 136
  diskette I/O 86
  DTFCD macro 59
  DTFDI macro 75
  DTFDU macro 86
  DTFIS macro 95
  DTFMR macro 101
  DTFMT 108
  DTFMT macro 108
  DTFOR macro 117
  DTFPR macro 129
  DTFSD macro 136
  indexed sequential file 95
  MICR/OCR input 101
  print output 129
  tape I/O 108
  1287 input 117
IDLOC operand (DTFDA) 68
IJBPUB (unit information DSECT) macro 171
IMAGE operand (DLINT) 49
increment document 40
index specification
  area (INDAREA of DTFIS) 93

high, unit of (HINDEX of DTFIS) 92
master (MSTIND of DTFIS) 97
size (INDSIZE of DTFIS) 93
skip (INDSKIP of DTFIS) 93
indexed sequential file
group retrieval 238
indexed-sequential I/O DTF 88
module for 175
read from by key (READ) 226
read from, sequential (GET) 166
sequential access, start of 238
set load mode (SETFL) 236
wait for end of I/O 258
write to by key (WRITE) 259
INPSIZE operand (DTFCN) 64
insert records 96
interval timer
See also IT exit routine
event control block (TECB) 252
set the (SETIME) 237
interval timer interrupt 247
IOSIZE operand (DTFIS) 96
ISMOD macro 175
IT exit routine
return from 149



JDUMP macro 179
job name
CPCLOSE argument list 41
job-communication (JOBCOM) macro 180
job, end of (EOJ) 146
journal tape processing
blocking line records 111, 204
device specification for 115, 204
online correction 225
record length for 117



KEEP operand
DTL macro 140
GENDTL macro 161
MODDTL macro 199
key argument (KEYARG) operand
direct access file 68
DTFDA macro 68
DTFIS macro 97
indexed sequential file 97
key length (KEYLEN) operand
direct access file 69
DTFDA macro 69
DTFIS macro 97
indexed sequential file 97
key location (KEYLOC) operand
(DTFIS) 97



label address (LABADDR) operand
direct access file (DTFDA) 69
disk sequential I/O (DTFSD) 137
physical IOCS file (DTFPH) 121
tape file (DTFMT) 108
label formats
disk volume 282
disk-user, standard 290
diskette volume 291
diskette, file standard 292
format-1 284
format-3 286
format-4 287
tape, file, standard 295
tape, user-standard file 298
tape, volume 294
label routine
coding a 279
direct access file (DTFDA) 69
disk sequential I/O (DTFSD) 137
physical IOCS file (DTFPH) 121
tape file (DTFMT) 108
label-extent information 72
label-identification code 279
labels, user
delete format-1 134
extent checking, direct access file 72
extent checking, physical IOCS 123
header information (on tape) 107, 121
identification code 279
non-standard, processing of 281
physical-IOCS file 121
processing of 279
return control (LBRET) 181
routine for, direct access file 69

routine for, disk sequential I/O 137
routine for, file on tape 108
tape, non-standard 298
trailer, for direct-access file 71
type of on tape 107
LBRET macro 181
length check (LENCHK) operand (DTFMT) 108
LENGTH operand
FREEVIS macro 159
GENDTL macro 161
GENIORB macro 164
GETVIS macro 169
I/O request block (GENIORB) 164
job communication data 181
JOBCOM macro 181
lock file (GENDTL) 161
virtual storage request 169
level test, subsystem 251
LFCB macro 183
LFR operand (DLINT) 49
LINBEG operand (DLINT) 49
line format (DLINT) 49
line-group definition (DFR) 44
lines-per-inch setting 184
LIOCS module, define a
card I/O (CDMOD) 21
device independent (DIMOD) 46
diskette I/O (DUMODFx) 142
IJMxxxxx (1287/1288 input) 205
indexed sequential file (ISMOD) 175
MICR/OCR file (MRMOD) 200
print output (PRMOD) 215
3886 file (DRMOD) 52
LITE macro 185
load
phase, retaining control 186
load forms control buffer (LFCB) macro 183
load mode
end of 145
file definition for 95
load phase (LOAD) macro 186
load request
See phase, load a
local directory list
address of 165
define a (GENL) 165
format definition (in FETCH) 156
format definition (in GENL) 166
format definition (in LOAD) 188
scan of by FETCH 156
scan of by LOAD 187
search of only (TXT of LOAD) 189
local time 168
lock a resource (LOCK) macro 190
lock control
control block for (DTL) 140
control block for (GENDTL) 161
system action summary 192
unlock resource (UNLOCK) 255
lock control block
assembly of, in program 140
define dynamically 161
modify a (MODDTL) 198
lock option (LOCKOPT) operand
DTL macro 141
GENDTL macro 162
MODDTL macro 199
logical unit (LOGUNIT) operand
GENIORB macro 164
logical unit (programmer), release of 228
lower/upper-case control 38
LPI operand (LFCB) 184

M

macro
coding fields of 1
definition of 1
notation 2
operands 1
register notation in 3
macro-format generate (MFG) operand
ATTACH macro 9
FETCH macro 157
GETIME macro 168
LOAD macro 189
PDUMP macro 209
STXIT macro 249
SUBSID macro 251
XECBTAB macro 263
magnetic ink character reader
See MICR/OCR device
map subsystem information (MAPSSID) macro 196
MAPBDY macro 194
MAPBDYVR macro 195
mark document/line 40
MFG operand

See macro-format generate (MFG) operand
MIC operand (GETIME) 168
MICR document buffer 27
MICR/OCR device
character set options (3886 file) 45
control operation for 40
document field, display of 53
dual address adapter 101
format/line-group definition (3886 file) 44
line format (3886 file) 49
pocket-lights control 185
read from (READ) 226
selective reread (RESCN) 231
sort mode, 1419 102
wait for end of I/O 258
write to (WRITE) 259
1255 file, define a (DTFMR) 100
1270/1275 file, define a (DTFMR) 100
1287/1288 file module (ORMOD) 204
1287/1288 file, define a (DTFOR) 111
1419 file, define a (DTFMR) 100
1419 sort mode 102
3886 file, define a (DTFDR) 78
3886 I/O module, define a 52
MODE operand (DTFCD) 60
modify lock control block (MODDTL) macro 198
module name (MODNAME) operand
card I/O 60
console I/O 64
diskette I/O 86
DTFCD macro 60
DTFCN macro 64
DTFDI macro 75
DTFDR macro 81
DTFDU macro 86
DTFIS macro 97
DTFMR macro 102
DTFOR macro 117
DTFPR macro 129
IJCxxxxx (card file) 23
IJDxxxxx (print output) 218
IJHxxxxx (indexed-sequential I/O) 178
IJJxxxxx (device independent) 47
IJMxxxxx 205
IJMZxxxx (3886 file) 53
IJNDxxxx (diskette file) 143
IJUxxxxx (MICR/OCR input) 200
indexed sequential file 97
MICR/OCR input 102
print output 129
1287/1288 input 117
3886 input 81
module, LIOCS, sub/supersetting
card I/O (IJCxxxxx) 24
device independent (IJJxxxxx) 48
diskette I/O (IJNDxxxx) 143
IJDxxxxx (print output) 219
IJHxxxxx 178
3886 file (IJCxxxxx) 53
MOUNTED operand (DTFPH) 122
move to communication region (MVCOM) 201
MRMOD macro 200
MSTIND operand (DTFIS) 97
MVCOM macro 201


N


NAME operand
ATTACH (subtask) macro 9
NATNHP operand (DFR) 46
non-standard labels, processing of 279
non-standard user label 298
NOSCAN operand (DLINT) 49
notation
conventions of 2
macros, general 2
operand 4
register 3
note record address (NOTE) macro 202
NRECDS operand (DTFIS) 97
NULMSG operand (LFCB) 184
numeric hand printing 46


O


OC exit routine
return from 149
open a file (OPEN, OPENR) macro 203
operand
keyword 2
positional 1

operand notation 4
operator communication
   See OC exit routine
optical character reader
   See MICR/OCR device
optical reader module (ORMOD)
 macro 204
OPTION operand (STXIT) 248
ORMOD macro 204
OUBLKSZ operand (DTFCD) 60
overflow control 220
OWNER operand
   DTL macro 141
   GENDTL macro 163
   MODDTL macro 200

P

page boundary
   load at (CDLOAD) 20
page control
   force page out 152
   page fault appendage (SETPFA) 240
   page-fix request (PFIX) 209
   page-free request (PFREE) 211
   page-in request (PAGEIN) 206
   release a page (RELPAG) 229
page fault appendage
   coding requirements 240
   linkage for (SETPFA) 240
   page fault queue in 241
   processing by 241
page fault queue, internal 243
PAGE operand (GETVIS) 169
page overflow
   See print overflow
page release request (RELPAG)
 macro 229
page-fix
   list 209
   request (PFIX) macro 209
page-free
   list 211
   request (PFREE) macro 211
page-in
   list 206
   request (PAGEIN) macro 206
page-overflow exit 221
page-release list 229
parameter specification
   CALL macro 10
partition
   activate (TPOUT) 253
   deactivate (TPIN) 253
partition boundary DSECT 194, 195
partition boundary information
   format of 195, 196
   request return of 150
PC exit routine
   return from 149
PDUMP macro 208
PFIX operand (GETVIS) 170
phase-name specification
   CDLOAD macro 20
phase, load a
   into partition GETVIS area
    (CDLOAD) 19
   local directory scan (FETCH) 156
   local directory scan (LOAD) 187
   passing control (FETCH) 155
   retain control unconditionally 189
   retaining control (LOAD) 186
   system directory scan (FETCH) 156
   system directory scan (LOAD) 188
phiocs.define file for (DTFPH) 119
physical IOCS
   access request (EXCP) 148
   command control block for 11
   disk I/O with RPS 234
   I/O request block (IORB) 163, 172
   sector value (SECTVAL) 234
   wait for end of I/O (WAIT) 256
physical write on FBA 137
PIOCS
   See physical IOCS
pocket-lights control 185
point macros
   behind noted record (POINTW) 213
   to noted record (POINTR) 212
   to start of file (POINTS) 213
POINTRG operand
   XPOST macro 265
   XWAIT macro 266
POOL operand (GETVIS) 170
position-file control
   note record address (NOTE) 202
   point behind noted record
    (POINTW) 213
   point to noted record (POINTR) 212
   point to start of file
    (POINTS) 213
post ECB (POST) macro 214
precision (PREC) operand (SETIME) 238

print file (VM) release (CPCLOSE) macro 41
print output definition
print overflow
  operand (PRINTOV of DTFPR) 129
  operand (PRINTOV of PRMOD) 217
printer
  character, table reference 130, 218
  control characters, ASA 269, 271
  control characters, EBCDIC 272
  control of, defining 126, 215
  data check control 131
  device control 37
  file, define a (DTFPR) 124
  output module (PRMOD) 215
  overflow (PRTOV) macro 220
  page overflow on 129, 217
  selective tape listing 130, 217
  table reference character 130, 218
printer-carriage spacing 37
priority, change for a task 25
private subtasks 240
program check
  See PC exit routine
program check interrupt 247
program, calling a 9
programmer logical unit, release of 228
PRTOV macro 220
publication, new in xi
publications, related iii
punch file (VM) release (CPCLOSE) macro 41
PUT macro 221
PUTR macro 223
PWRITE operand (DTFSD) 137




random retrieval
  See indexed sequential file
RC operand (DUMP) 145
RCB, define a 224
rcbname specification
  DEQ macro 42
reactivate partition (TPOUT) 253
read a line (RDLNE) macro 225
read from (1287) keyboard 40
READ macro 226
read mode, direct access 69
READ operand (DTFMT) 108
read request
  sequential (GET) 166
read-only module
  card I/O 22, 60
  device independent 47
  diskette I/O 86
  diskette I/O module 142
  DRMOD macro 52
  DTFCD macro 60
  DTFDI macro 76
  DTFDR macro 81
  DTFDU macro 86
  DTFIS macro 98
  DTFPR macro 129
  DUMODFx macro 142
  indexed sequential file 98
  indexed sequential I/O module 177
  ISMOD macro 177
  print output 129
  print output module 217
  PRMOD macro 217
  3886 input 81
READID operand (DTFDA) 69
READKEY operand (DTFDA) 69
real address (in CCW) 148
real address return (REALAD) macro 227
record format (RECFORM) operand
  card I/O 61
  console I/O 64
  direct access file 69
  disk sequential I/O 137
  DTFCD 61
  DTFCN macro 64
  DTFDA macro 69
  DTFIS 98
  DTFPR macro 130
  DTFSD macro 137
  indexed-sequential file 98
  indexed-sequential I/O module 177
  ISMOD macro 177
  print output 130
  print output module 217
  PRMOD macro 217
  tape I/O 108
  1287/1288 input 117
  1287/1288 input module 204
record length
  device-independent file 76
  diskette-I/O file 87
  journal tape processing 117

MICR/OCR file (DTFMR) 102
tape I/O 109
undefined, card I/O 61
undefined, console I/O 64
undefined, direct access 70
record length, print output 125
record size (RECSIZE) operand
card I/O 61
console I/O 64
direct access file 70
disk sequential I/O 137
diskette I/O 87
DTFCD macro 61
DTFCN macro 64
DTFDA macro 70
DTFDI macro 76
DTFDU macro 87
DTFIS macro 98
DTFMR macro 102
DTFMT macro 109
DTFOR macro 117
DTFPR macro 130
DTFSD macro 137
indexed-sequential file 98
MICR/OCR input 102
print output 130
tape I/O 109
1287/1288 input 117
records, number of (DTFIS) 97
register
notation 3
saving of (SAVE) 233
REJECT operand (DFR) 46
related publications iii
release
block of data (RELSE) 231
disk area (FREE) 158
page, virtual storage 229
programmer logical unit 228
resource (DEQ) 42
virtual storage (FREEVIS) 159
release block (RELSE) macro 231
release logical unit (RELEASE) macro 228
release page (RELPAG) macro 229
RELTYPE operand (DTFDA) 70
RESCN macro 231
resource control
dequeue an RCB 42
enqueue RCB (ENQ) 145
lock a (LOCK) 190
unlock a (UNLOCK) 255
resource control block
CCB 11
DTFCD 54
DTFCN 63
DTFDA 66
DTFDI 73
DTFDR 78
DTFDU 82
DTFIS 88
DTFMR 100
DTFMT 103
DTFOR 111
DTFPH 119
DTFPR 124
DTFSD 132
event control block (ECB) 8
I/O request
See I/O request block (IORB)
IORB 172
lock control
See lock control block
RCB 224
XECBTAB 261
resource control block (RCB) macro 224
restart
disk address for 33
end address 33
instruction-address for 32
tape position for 33
restart address (CHKPT) 32
RET operand (LOAD) 189
RETPNF operand (CDLOAD) 20
retrieve records 96
return control
interrupt exit (EXIT) 149
label processing (LBRET) 181
return control (RETURN) macro 232
rewind tape 37
on OPEN 109
programmed request (CNTRL) 37
rewind tape (REWIND) operand (DTFMT) 109
rotational position sensing 177
rotational position sensing (RPS) 234
RPS operand (ISMOD) 177
RUNMODE macro 233

S


save area, subtask 8
SAVE operand
  ASSIGN macro 6
  ATTACH macro 7
  DETACH macro 43
  POST macro 214
save register (SAVE) macro 233
SCOPE operand
  DTL macro 141
  GENDTL macro 163
  MODDTL macro 200
search multiple tracks 71
SECADDR operand (DTFMR) 102
secondary control unit (MICR input) 102
sector value (SECTVAL) macro 234
seek address 36
SEEKADR operand (DTFDA) 70
selective online correction 225
selective reread (RESCN) 231
selective tape list feature 130, 217
selective tape listing 222
sense-CCW specification (CCB) 11
separate assembly
  CDMOD macro 23
  device independent 47
  disk sequential I/O 138
  diskette I/O 87
  diskette I/O module 143
  DRMOD macro 53
  DTFCD macro 61
  DTFDA macro 71
  DTFDI macro 77
  DTFDR macro 81
  DTFDU macro 87
  DTFIS macro 98
  DTFMR macro 102
  DTFMT macro 109
  DTFOR macro 118
  DTFPR macro 130
  DTFSD macro 138
  DUMODFx macro 143
  indexed-sequential file 98
  indexed-sequential I/O module 177
  ISMOD macro 177
  MICR/OCR input 102
  MICR/OCR input module 201
  MRMOD macro 201
  ORMOD macro 204
  print output 130
  print output module 217
  PRMOD macro 217
  tape I/O 109
  1287/1288 input 118
  1287/1288 input module 204
  3886 input 81
sequential access
  end of (ESETL) 147
  start of 238
sequential file
  card I/O DTF 54
  card I/O module 21
  console I/O DTF 63
  device independent module 46
  device-independent I/O DTF 73
  disk I/O DTF 132
  diskette I/O DTF 82
  diskette I/O module 142
  print output DTF 124
  read from (GET) 166
  release block (RELSE) 231
  tape DTF 103
  write to (PUT) 221
  1255 input DTF 100
  1270/1275 input DTF 100
  1287/1288 input DTF 111
  1419 input DTF 100
  3886 input DTF 78
  3886 input module 52
set device (SETDEV)
  operand (DRMOD) 53
  operand (DTFDR) 82
set exit (STXIT) 245
set file load mode (SETFL) macro 236
set interval timer (SETIME) macro 237
set limits (SETL) macro 238
set page fault appendage (SETPFA) macro 240
set task timer (SETT) macro 244
SETDEV macro 236
short block
  See truncation of data
sort mode, 1419 102
SORTMDE operand (DTFMR) 102
spanned records
  skipping of (by CNTRL) 37
  spacing over 231
SPID operand
  FREEVIS macro 160
  GETVIS macro 170
spool file (VM) release (CPCLOSE) macro 41

SRCHM operand (DTFDA) 71
SSELECT operand (DTFCD) 61
stacker eject/selection
  card I/O 39
  card I/O (SSELECT of DTFCD) 61
  control characters, ASA 269
  control characters, EBCDIC 271
  document I/O 40
standard label
  IBM file, disk 283
  IBM file, diskette 292
  IBM file, tape 295
  IBM volume, diskette 291
  IBM volume, tape 294
  processing of 279
  user file, disk 289
  user file, tape 298
STANDARD operand (GETIME) 167
STLIST operand
  DTFPR macro 130
  PRMOD macro 217
STLSK operand (PUT) 222
STLSP operand (PUT) 222
storage control
  See virtual storage control
storage dump
  See dump request by
STXIT routine
  interrupt in 250
  set up an 245
subroutine
  See exit routine
subsystem information
  level test 251
  mapping DSECT for 196
  request (SUBSID) macro 251
subtask
  See task
SVA operand
  FREEVIS macro 160
  GETVIS macro 170
synchronize tape buffer 37
SYS operand
  FETCH macro 156
  LOAD macro 188
system directory list
  scan of by FETCH 156
  scan of by LOAD 188
  search of only (TXT of LOAD) 189
system end-of-volume (SEOV) macro 235
SYSxxx specification
  CCB macro 11
  CHKPT macro 32
  forms control buffer load device 183
  LFCB macro 183
  RELEASE macro 228

T

table reference character 47, 77, 130, 218
tape buffer, synchronization of 37
tape device
  dynamic assign of 5
  restart position (CHKPT) 33
tape file, magnetic
  define a (DTFMT) 103
  device control 37
  end of volume SYSLST/SYSPCH (SEOV) 235
  force end of (FEOV) 154
  physical IOCS file (DTFPH) 119
  release block (RELSE) 231
  SYSLST/SYSPCH end of volume (SEOV) 235
tape label
  file, standard 295
  file, user-standard 298
  non-standard 298
  standard, file 295
  standard, volume 294
  volume 294
tape mark
  write a (CNTRL) 37
  write a (DTFMT) 110
task
  attach a (ATTACH) 6
  cancel a 10
  deactivate a (DETACH) 43
  privately controlled 240
  save area for 8
  set low priority for 25
  timer
    See TT exit routine
  timer, set the (SETT) 244
  timer, testing the (TESTT) 252
  wait on programmed event 256
task subpool 170
task-to-task communication
  dequeue resource (DEQ) 42
  enqueue resource (ENQ) 145
  event control block for 8

post ECB (POST) 214
TECB macro 252
telecommunication
See communication
test interval timer (TTIMER) 254
test task timer (TESTT) macro 252
time of day request (GETIME) macro 167
timer
interval, event control block (TECB) 252
interval, setting of (SETIME) 237
interval, testing the (TTIMER) 254
task, set the (SETT) 244
task, testing the (TESTT) 252
TPIN macro 253
TPOUT macro 253
trailer labels, direct access file 71
translation, ASCII to EBCDIC to ASCII 275
transmission information, CCB 14
TRC (table reference character) operand
DTFPR macro 130
PRMOD macro 218
TRC operand (DIMOD) 47
TRC operand (DTFDI) 77
TRLBL operand (DTFDA) 71
truncate (TRUNC) macro 254
truncation of data
disk sequential I/O 138
request macro (TRUNC) 254
TRUNCS operand, DTFSD macro 138
TSKSUBP operand (GETVIS) 170
TT exit routine
return from 149
TTIMER macro 254
TU operand (GETIME) 168
TXT operand (LOAD) 189
type of file (TYPEFLE) operand
card I/O 62
CDMOD macro 23
console I/O 64
DIMOD macro 47
direct access file 71
disk sequential I/O 138
diskette I/O 87
DTFCD macro 62
DTFCN macro 64
DTFDA macro 71
DTFDU macro 87
DTFIS macro 98
DTFMT macro 110
DTFPH macro 123
DTFSD macro 138
indexed-sequential file 98
indexed-sequential I/O module 177
ISMOD macro 177
physical IOCS 123
tape I/O 110
TYPE operand (XECBTAB) 261



UCS operand (DTFPR) 131
unit information
request return of 150
unit record device
dynamic assign of 5
unit-information DSECT (IJBPUB) macro 171
unload tape
on CLOSE 109
programmed request (CNTRL) 37
unlock resource (UNLOCK) macro 255
update a file
indexed sequential 96
UPDATE operand (DTFSD) 138
upper/lower-case control 38
user label
non-standard, processing of 281
processing of 279
standard, format of 289
user labels
See labels, user



variable-length record build (VARBLD) operand
disk sequential I/O 138
DTFMT macro 110
DTFSD macro 138
tape I/O 110
VERIFY operand
direct access file 71
disk sequential I/O 139
diskette I/O 87
DTFDA macro 71
DTFDU macro 87

DTFIS macro 99
DTFSD macro 139
indexed-sequential file 99
virtual address return (VIRTAD) macro 256
virtual storage control
force page out 152
free virtual storage (FREEVIS) 159
get virtual storage (GETVIS) 169
page fault appendage (SETPFA) 240
page-fix request (PFIX) 209
page-free request (PFREE) 211
page-in request (PAGEIN) 206
real address return (REALAD) 227
release a page (RELPAG) 229
run mode notification 233
virtual address return 256
virtual storage release
address of area 159
from subpool 160
from SVA 160
length of area 159
macro (FREEVIS) 159
virtual storage request
fixed area 170
from pool 170
from system GETVIS 170
GETVIS macro 169
page boundary allocation 169
subpool for 170
VM spool file release (CPCLOSE) macro 41
VOLSEQ operand (DTFDU) 87
volume identifier (VOLID) operand
DTL macro 142
GENDTL macro 163
MODDTL macro 200
volume label
disk 282
volume, force end of
disk (FEOVD) 155
SYSLST/SYSPCH on tape (SEOV) 235
tape (FEOV) 154

## W

wait state
beginning of for a task (WAIT) 256
ending of for a task 214
file access (WAITF) 258
MICR file read (CHECK) 26
multiple events (WAITM) 258
work file read/write (CHECK) 26
WLRERR operand
device independent I/O 77
disk sequential I/O 139
DTFDI macro 77
DTFMT macro 110
DTFSD macro 139
tape I/O 110
work area specification
CDMOD macro 23
console I/O 65
disk sequential I/O 139
diskette I/O 87
DTFCD macro 62
DTFCN macro 65
DTFDU macro 87
DTFMT macro 111
DTFOR macro 118
DTFPR macro 131
DTFSD macro 139
indexed sequential file 99
ORMOD macro 205
print output 131
print output module 218
PRMOD macro 218
read request (GET) 166
tape I/O 111
write request (PUT) 221
write request (PUTR) 223
1287/1288 input 118
1287/1288 input module 205
work file
disk sequential I/O 138
point behind noted record (POINTW) 213
point to noted record (POINTR) 212
point to start of file (POINTS) 213
read from (READ) 226
tape 110
wait for end of I/O (CHECK) 26
WORKL operand (DTFIS) 99
WORKR operand (DTFIS) 99
WORKS operand (DTFIS) 99
WRITE macro 259
write protect (on diskette) 87
write tapemark
on CLOSE (DTFMT) 110
programmed request (CNTRL) 37
write to file
by record ID 71

by record key 72
with formatting 66
WRITEID operand (DTFDA) 71
WRITEKY operand (DTFDA) 72
wrong-length-record exit routine
device independent I/O 77
disk sequential I/O 139
tape I/O 110
WRTPROT operand (DTFDU) 87

## X

XECB operand
XECBTAB macro 262
XPOST macro 265
XWAIT macro 266
XECB table entry
access type 263
check a 261
define a 261
delete a 261
name of 262
post a (XPOST) 264
reset a 262
wait on (XWAIT) 266
XPOST macro 264
XTNTXIT operand (DTFDA) 72
XWAIT macro 266

## Numerics

1255
See MICR/OCR device
1287
See MICR/OCR device
1288
See MICR/OCR device
1403
See printer
1419
See MICR/OCR device
1442
See card I/O
1443
See printer
2311
See disk device
2314/2319
See disk device
2400 series tape
See tape file, magnetic
2520
See card I/O
2540
See card I/O
2560
See card I/O
3203
See printer
3211
See printer
3330/3333
See disk device
3340/3344
See disk device
3350
See disk device
3375
See disk device
3380
See disk device
3400 series tape
See tape file, magnetic
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
5424
See card I/O
5425
See card I/O
8809
See tape file, magnetic

IBM Virtual Storage Extended/
Advanced Functions
Application Programming
Macro Reference
Order No. SC33-6197-02

# READER'S COMMENT FORM

This form may be used to communicate your views about this publication. They will be sent to the author's department for whatever review and action, if any, is deemed appropriate. Comments may be written in your own language; use of English is not required.
IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

**Note:** *Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.*

Possible topics for comments are:

Clarity Accuracy Completeness Organization Coding Retrieval Legibility

If you wish a reply, give your name and mailing address: _____

_____

_____

_____

What is your occupation? _____

Number of latest Newsletter associated with this publication: _____

Thank you for your cooperation. No postage stamp is necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page).

Note: Staples can cause problems with automated mail sorting equipment. Please use pressure sensitive or other gummed tape to seal this form.

Cut or Fold Along Line

SC33-6197-02

**Reader's Comment Form**

Fold and Tape **Please Do Not Staple** Fold and Tape

No Postage necessary if mailed in the United States

**BUSINESS REPLY MAIL**

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

Postage will be paid by addressee:

**International Business Machines Corp.**
**Department 6R1BP**
**180 Kost Road**
**Mechanicsburg, PA 17055**

Fold and Tape **Please Do Not Staple** Fold and Tape

IBM®

IBM Virtual Storage Extended/
Advanced Functions
Application Programming
Macro Reference
Order No. SC33-6197-02

# READER'S COMMENT FORM

This form may be used to communicate your views about this publication. They will be sent to the author's department for whatever review and action, if any, is deemed appropriate. Comments may be written in your own language; use of English is not required.
IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

**Note:** *Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.*

Possible topics for comments are:

Clarity Accuracy Completeness Organization Coding Retrieval Legibility

If you wish a reply, give your name and mailing address: _____

_____

_____

_____

What is your occupation? _____

Number of latest Newsletter associated with this publication: _____

Thank you for your cooperation. No postage stamp is necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page).

Note: Staples can cause problems with automated mail sorting equipment. Please use pressure sensitive or other gummed tape to seal this form.

Cut or Fold Along Line

SC33-6197-02

**Reader's Comment Form**

Fold and Tape | **Please Do Not Staple** | Fold and Tape

No Postage necessary if mailed in the United States

**BUSINESS REPLY MAIL**
FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

Postage will be paid by addressee:

**International Business Machines Corp.**
**Department 6R1BP**
**180 Kost Road**
**Mechanicsburg, PA 17055**

Fold and Tape | **Please Do Not Staple** | Fold and Tape

IBM®

IBM Virtual Storage Extended/
Advanced Functions
Application Programming
Macro Reference
Order No. SC33-6197-02

# READER'S COMMENT FORM

This form may be used to communicate your views about this publication. They will be sent to the author's department for whatever review and action, if any, is deemed appropriate. Comments may be written in your own language; use of English is not required.
IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

**Note:** *Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.*

Possible topics for comments are:

Clarity Accuracy Completeness Organization Coding Retrieval Legibility

If you wish a reply, give your name and mailing address: _____

_____

_____

_____

What is your occupation? _____

Number of latest Newsletter associated with this publication: _____

Thank you for your cooperation. No postage stamp is necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page).

Note: Staples can cause problems with automated mail sorting equipment. Please use pressure sensitive or other gummed tape to seal this form.

Cut or Fold Along Line

SC33-6197-02

**Reader's Comment Form**

Fold and Tape | **Please Do Not Staple** | Fold and Tape

No Postage necessary if mailed in the United States

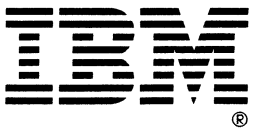
**BUSINESS REPLY MAIL**
FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

Postage will be paid by addressee:

**International Business Machines Corp.**
**Department 6R1BP**
**180 Kost Road**
**Mechanicsburg, PA 17055**

Fold and Tape | **Please Do Not Staple** | Fold and Tape

IBM®



IBM Virtual Storage Extended/
Advanced Functions
Order Number SC33-6197-02
(File No. S370/4300-30)
Printed in U.S.A.

SC33-6197-02

