File No. S360-20
Order No. GC28-6550-11

OS

Systems Reference Library

OS Data Management for System Programmers

Release 21

This publication consists of self-contained chapters, each of which provides information on how to modify, extend, or implement the data management capabilities of the IBM System/360 Operating System control program. It is designed primarily for system programmers responsible for maintaining, updating, and extending the operating system features.

Topics:

Catalog and VTOC Maintenance
IECDSECT, IEFJFCBN, and IEFUCBOB Macro Instructions
The EXCP Macro Instruction
The XDAP Macro Instruction
Implementing Data Set Protection
Adding a UCS Image to the System Library

Twelfth Edition (April 1973)

This edition replaces the previous edition (numbered GC28-6550-10) and its technical newsletter (numbered GN26-0750) and makes them both obsolete.

This edition applies to release 21.7 and to all subsequent releases unless otherwise indicated in new editions or technical newsletters.

Significant changes are summarized under "Summary of Amendments" following the list of illustrations. Each technical change is marked by a vertical line to the left of the changed area.

Information in this publication is subject to significant change. Any such changes will be published in new editions or technical newsletters. Before using the publication, consult the latest *IBM System/360 and System/370 Bibliography*, GA22-6822, and the technical newsletters that amend that bibliography, to learn which editions and technical newsletters are applicable and current.

Requests for copies of IBM publications should be made to the IBM branch office that serves you.

Forms for reader's comments are provided in the back of this publication. If the forms have been removed, comments may be addressed to IBM Corporation, Programming Center–Publishing, Department D58, Monterey and Cottle Roads, San Jose, California 95193. All comments become the property of IBM.

© Copyright International Business Machines Corporation 1966, 1967, 1968, 1969, 1970, 1971, 1972, 1973

Preface

This publication consists of self-contained chapters, each of which provides system programmers with information on how to modify, extend, or implement the data management capabilities of the IBM System/360 Operating System control program. Although the information in one chapter is sometimes related to information in another, all chapters have been written as separate and complete units. It is assumed that users of this publication are thoroughly familiar with the design of the operating system and its features. Such information can be obtained in IBM System/360 Operating System: Introduction, GC28-6534. Each chapter contains its own introductory section and list of prerequisite publications. This organization has been used to reduce cross-referencing.

Contents

Figures

TAPE DRIVE FEATURE SUPPORT

The 7-track feature is now supported on all 3400 model tape drives. The UCB Type Field has been corrected for 3400 tape drives, in the DEVTYPE macro instruction portion of the manual.

MISCELLANEOUS CHANGES

Other technical and editorial corrections have been made throughout the manual.

Summary of Amendments
for GC28-6550-10
OS Release 21

NEW DEVICE SUPPORT

Information is added to support the IBM 3803/3420 Magnetic Tape Subsystem and the IBM 3505/3525 Card Reader/Card Punch.

CATALOG MANAGEMENT

Changes have been made to the section on catalog and VTOC maintenance to show changes in the method of adding, deleting and naming data sets in the catalog.

MACRO INSTRUCTIONS ADDED

Information is added about the CVT and LABEL macro instructions.

ORGANIZATION CHANGE

Information from the following chapters has been relocated to the IBM System/360 Operating System MFT Guide, GC27-6939:

- The Must Complete Function
- Job Queue Formatting
- The PRESRES Volume Characteristics List
- Output Separation
- Writing System Output Writer Routines
- Adding SVC Routines to the Control Program
- Message Routing Exit Routines
- Handling Accounting Routines
- Reader/Interpreter and Output Writer Cataloged Procedures
- Resident Routines Option
- The Shared DASD Option
- The Time Slicing Facility
- System Macro Instructions (except DEVTYPE, OPEN, and RDJFCB)

Information from the following chapters has been relocated to the IBM System/360 Operating System MVT Guide, GC28-6720:

- The Must Complete Function
- Job Queue Formatting
- The PRESRES Volume Characteristics List
- Output Separation
- Writing System Output Writer Routines
- Adding SVC Routines to the Control Program
- Message Routing Exit Routines
- Handling Accounting Routines
- Reader/Interpreter and Output Writer Cataloged Procedures - Dedicated Data Sets
- Using the Link Pack Area
- Writing Rollout/Rollin Installation Appendages
- The Shared DASD Option
- The Time Slicing Facility
- System Macro Instructions (except DEVTYPE, OPEN, and RDJFCB)

The chapter about graphic job processing has been relocated to the User's Guide for Job Control from the IBM 2250 Display Unit, GC27-6933.

The chapter about satellite graphic job processing has been relocated to the User's Guide for Job Control from the IBM 2250 Display Unit Attached to an IBM 1130 System, GC27-6938.

The chapter about System Management Facilities was removed during the last revision. The information may now be found in IBM System/360 Operating System SMF Guide, GC28-6715.

Information about the tracing routine option may now be found in IBM System/360 Operating System: Programmer's Guide to Debugging, GC28-6670.

MISCELLANEOUS CHANGES

Information is added to clarify specifications of the password data set and the use of the IEHPROGM utility program for updating the password data set.

New Event Control Block (ECB) codes have been added to the descriptions in the EXCP and XDAP macro instruction sections.

Item	Description	Chapter Affected
TSO	The PURGE parameter list has a fourth word that can be used to purge a list of TCBs.	Execute Channel Program
START command	The START command can now be used to start a problem program.	System Reader, Initiator, and Writer Cataloged procedures
7094 Emulator	Change to the ASB procedure for 7094 Emulator	System Reader, Initiator, and Writer Cataloged Procedures
FORTRAN G	Change to data blocking for FORTRAN G	System Reader, Initiator, and Writer Cataloged Procedures
STAE	Change to STAE retry routine procedure	System Macro Instructions
PROTECT	Additional return code for PROTECT macro instruction	Data Set Protection
3211 Printer	New device dependent information for the 3211 Printer	IECDSECT, IEFJFCBN, and IEFUCBOB Macro Instructions Execute Direct Access Program (XDAP) Macro Instructions System Macro Instructions Writing System Output Writer Routines Output Separation System Reader, Initiator, and Writer Cataloged procedures Adding a Universal Character Set Image or FCB Image to the Image Library
3330 and 2305 Direct Access Devices	New device dependent information for the 3330 and 2305 Direct Access Devices	Maintaining the Catalog and the Volume Table of Contents Execute Direct Access Program (XDAP) Macro Instruction System Macro Instructions The Shared Direct Access Storage Device Option

Maintaining the Catalog and the Volume Table of Contents

This chapter provides detailed information on how to maintain and modify the catalog and volume table of contents.

Before reading this chapter, you should be familiar with the information contained in the prerequisite publications listed below.

PREREQUISITE PUBLICATIONS

The IBM System/360 Operating System: Assembler Language publication (GC28-6514) contains the information necessary to code programs in the assembler language.

The IBM System/360 Operating System: Data Management Services publication (GC26-3746) contains a general description of the structure of catalog indexes, as well as a brief discussion of the volume table of contents (VTOC).

COMPANION PUBLICATION

The IBM System/360 Operating System: System Control Blocks publication (GC28-6628) contains format and field descriptions of the system control blocks referred to in this chapter.

RECOMMENDED PUBLICATIONS

The IBM System/360 Operating System: Utilities publication (GC28-6586) describes how to maintain and modify the catalog and the volume table of contents through the use of utility programs.

Maintaining the Catalog and the Volume Table of Contents

This chapter describes how to maintain and modify the catalog and the volume table of contents through the use of macro instructions. Most of the maintenance and modification functions can also be performed using utility statements. The utility statements are described in the publication IBM System/360 Operating System: Utilities.

The functions you can perform using the macro instructions are described in text, and the formats of the macro instructions are tabulated on a fold-out sheet (Figure CTLG1) at the back of this book. The chart on the fold-out sheet associates the function described in text with the macro instructions needed to perform the function. You should keep the fold-out sheet open when reading the text.

The functions that are described in text are:

- How to read a block from the catalog.
- How to build an index.
- How to build a generation index.
- How to delete an index.
- How to assign an alias.
- How to delete an alias.
- How to connect control volumes.
- How to disconnect control volumes.
- How to catalog a data set.
- How to remove data set references from the catalog.
- How to recatalog a data set.
- How to read a data set control block from the volume table of contents.
- How to delete a data set.
- How to rename a data set.

Accompanying the function descriptions in text are coding examples and programming notes; exceptional-return condition codes for the macro instructions are tabulated on the back of the fold-out sheet (Figure CTLG2). In the functional descriptions, bytes of data blocks are numbered from zero (the first byte is byte zero).

HOW TO READ A BLOCK FROM THE CATALOG

To read either an index block or a block indicating the volumes on which a data set is stored (volume-list block), you use the LOCATE and CAMLST macro instructions. There are two ways to specify the block that you want read into main storage: by using the name of the index level or data set, or by using the block's location relative to the beginning of the catalog (TTR).

-By Specifying the Name of a Data Set

If you specify an index level name, the first block of the named index is read into main storage, and an exceptional return code is set. Index block formats are contained in Appendix A of this chapter.

If you specify a data set name, a 256-byte volume-list block is read into main storage. The block contains up to 20 volume pointers, each of which points to a volume on which part of the data set is stored. The first two bytes of the block contain the number of volume pointers for the data set. Each volume pointer is a 12-byte field that contains a 4-byte device code, a 6-byte volume serial number, and a 2-byte data set sequence number. (Device codes are contained in Appendix B of this chapter.)

If the named data set is stored on more than 20 volumes, bytes 252-254 of the block contain the relative track address of the next volume-list block of volume pointers. Byte 255 contains a binary zero.

If the named data set is stored on only one volume, bytes 252-254 of the block contain the relative track address of the DSCB for that data set, otherwise these bytes are zero. Byte 255 contains a binary zero.

Example: In the following example, the list of volumes that contain data set A.B is read into main storage. The search for the volume-list block starts on the system residence volume.

```
| Name     | Operation | Operand                                          |
|----------|-----------|--------------------------------------------------|
|          | LOCATE    | INDAB                  READ VOLUME-LIST BLOCK FOR |
|      Check Exceptional Returns            CATALOGED DATA SET A.B INTO|
| INDAB    | CAMLST    | NAME,AB,,LOCAREA       MAIN STORAGE AREA NAMED    |
| AB       | DC        | CL44'A.B'              LOCAREA. LOCAREA ALSO      |
| LOCAREA  | DS        | 0D                     CONTAINS 3-BYTE TTR AND    |
|          | DS        | 265C                   6-BYTE SERIAL NUMBER       |
```

The LOCATE macro instruction points to the CAMLST macro instruction. NAME, the first operand of CAMLST, specifies that the system is to search the catalog for a volume-list block by using the name of a data set. AB, the second operand, specifies the main storage location of a 44-byte area into which you have placed the fully qualified name of a data set. LOCAREA, the fourth operand, specifies a 265-byte area you have reserved in main storage.

After execution of these macro instructions, the 265-byte area contains: the 256-byte volume-list block for the data set A.B and the 6-byte serial number of the volume on which the block was found (in bytes 259-264). If data set A.B resides on only one volume, bytes 252-254 of the volume-list block contain the relative track address of the DSCB for data set A.B (relative to the beginning of the volume).

If a code of 4 is returned in register 15 indicating that the required control volume was not mounted, bytes 259-264 of the work area will contain the volume serial number of this required volume. If LOCATE finds an old CVOL pointer entry, and the CVOL is not mounted, binary zeros will be returned in bytes 252-255 of the work area. However, if a new CVOL pointer entry is found, the four-byte device code of the CVOL will be returned in those bytes.

-By Specifying the Name of a Generation Data Set

You specify the name of a generation data set by using the fully qualified generation index name and the relative generation number of the data set. The value of a relative generation number reflects the position of a data set in a generation data group. The following values can be used:

- Zero - specifies the latest data set cataloged in a generation data group.
- Negative number - specifies a data set cataloged before the latest data set.
- Positive number - specifies a data set not yet cataloged in the generation data group.

When you use zero or a negative number as the relative generation number, a volume-list block is read into main storage and the relative generation number is replaced by the absolute generation name.

When you use a positive number as the relative generation number, an absolute generation name is created and replaces the relative generation number. A volume-list block is not read, since none exists for these data sets.

Example: In the following example, the list of volumes that contain generation data set A.PAY(-3) is read into main storage. The search for the volume-list block starts on the system residence volume.

```
┌─────────┬─────────┬──────────────────────────────────────────────┐
│ Name    │ Operation│ Operand                                     │
├─────────┼─────────┼──────────────────────────────────────────────┤
│         │ LOCATE  │ INDGX                 READ VOLUME-LIST BLOCK FOR │
│     Check Exceptional Returns           DATA SET A.PAY(-3) INTO  │
│ INDGX   │ CAMLST  │ NAME,APAY,,LOCAREA    MAIN STORAGE AREA NAMED  │
│ APAY    │ DC      │ CL44'A.PAY(-3)'       LOCAREA. LOCAREA ALSO CON- │
│ LOCAREA │ DS      │ 0D                    TAINS 3-BYTE TTR AND     │
│         │ DS      │ 265C                  6-BYTE SERIAL NUMBER     │
└─────────┴─────────┴──────────────────────────────────────────────┘
```

The LOCATE macro instruction points to the CAMLST macro instruction. NAME, the first operand of CAMLST, specifies that the system is to search the catalog for a volume-list block by using the name of a data set. APAY, the second operand, specifies the main storage location of a 44-byte area into which you have placed the name of the generation index and the relative generation number of a data set in the generation data group. LOCAREA, the fourth operand, specifies a 265-byte area you have reserved in main storage.

After execution of these macro instructions, the 265-byte area contains: the 256-byte volume-list block for generation data set A.PAY(-3) and the 6-byte serial number of the volume on which the block was found (in bytes 259-264). If data set A.PAY(-3) resides on one volume, bytes 252-254 of the volume-list block contain the relative track address of the DSCB for that data set (relative to the beginning of the volume). In addition, the system will have replaced the relative generation number that you specified in your 44-byte area with the data set's absolute generation name.

-By Specifying a Name Using an Alias

For each of the preceding functions, you can specify an alias as the first name in the qualified name of an index level, data set, or generation data set. Each function is performed exactly as previously described, with one exception: the alias name specified is replaced by the true name.

-By Specifying by TTR

You can read any block in the catalog by specifying, in the form TTR, the identification of the block and its location relative to the beginning of the catalog. TT is the number of tracks from the beginning of the catalog, R is the record number of the desired block on the track. (Formats of each type of catalog block are contained in Appendix A of this chapter.)

Example: In the following example, the block at the location indicated by TTR is read into main storage. The specified block is in the catalog on the system residence volume.

```
| Name     | Operation | Operand                                          |
|          | LOCATE    | BLK                    READ A BLOCK INTO MAIN    |
|    Check Exceptional Returns          STORAGE AREA NAMED LOCAREA |
| BLK      | CAMLST    | BLOCK,TTR,,LOCAREA                               |
| TTR      | DC        | H'5'                   RELATIVE TRACK 5          |
|          | DC        | X'03'                  BLOCK 3 ON TRACK          |
| LOCAREA  | DS        | 0D                     LOCAREA ALSO CONTAINS 3-BYTE|
|          | DS        | 265C                   TTR AND 6-BYTE SERIAL NO. |
```

The LOCATE macro instruction points to the CAMLST macro instruction. BLOCK, the first operand of CAMLST, specifies that the system is to search the catalog for the block indicated by TTR, the second operand. LOCAREA, the fourth operand, specifies a 265-byte area you have reserved in main storage.

After execution of these macro instructions, the 265-byte area contains the 256-byte index block and the 6-byte serial number of the volume on which the block was found (in bytes 259-264).

HOW TO BUILD AN INDEX

To build a new index structure and add it to the catalog, you must create each level of the index separately. (You can also create index levels while you are cataloging a data set onto those index levels. See "How to Catalog a Data Set" in this chapter for details.) You create each level of the index by using the INDEX and CAMLST macro instructions.

These two macro instructions can also be used to add index levels to existing index structures.

Example: In the following example, index structure A.B.C is built on the control volume whose serial number is 000045.

```
| Name     | Operation | Operand                                          |
|          | INDEX     | INDEXA                 BUILD INDEX A             |
|    Check Exceptional Returns                                          |
|          | INDEX     | INDEXB                 BUILD INDEX STRUCTURE A.B |
|    Check Exceptional Returns                                          |
|          | INDEX     | INDEXC                 BUILD INDEX STRUCTURE A.B.C|
|    Check Exceptional Returns                                          |
| INDEXA   | CAMLST    | BLDX,ALEVEL,VOLNUM                               |
| INDEXB   | CAMLST    | BLDX,BLEVEL,VOLNUM                               |
| INDEXC   | CAMLST    | BLDX,CLEVEL,VOLNUM                               |
| VOLNUM   | DC        | CL6'000045'            VOLUME SERIAL NUMBER      |
| ALEVEL   | DC        | CL2'A'                 INDEX STRUCTURE NAMES     |
| BLEVEL   | DC        | CL4'A.B'               FOLLOWED BY BLANKS        |
| CLEVEL   | DC        | CL6'A.B.C'             WHICH DELIMIT FIELDS      |
```

Each INDEX macro instruction points to an associated CAMLST macro instruction. BLDX, the first operand of CAMLST, specifies that an index level be built. The second operand specifies the main storage location of an area into which you have placed the fully qualified name of an index level. The third operand specifies the main storage location of an area into which you have placed the 6-byte serial number of the volume on which the index level is to be built.

HOW TO BUILD A GENERATION INDEX

You build a generation index by using the INDEX and CAMLST macro instructions. All higher levels of the index must exist. If the higher levels of the index are not in the catalog, you must build them. How to build an index has been explained previously. In the following example, the generation index D is built on the control volume whose serial number is 000045. The higher level indexes A.B.C already exist. When the number of generation data sets in the generation index D exceeds four, the oldest data set in the group is uncataloged and scratched.

```
| Name     | Operation | Operand                                          |
|          | INDEX     | GENINDX              BUILD GENERATION INDEX      |
|      Check Exceptional Returns                                          |
| GENINDX  | CAMLST    | BLDG,DLEVEL,VOLNUM,,DELETE,,4                    |
| DLEVEL   | DC        | CL8'A.B.C.D'                    BLANK DELIMITER  |
| VOLNUM   | DC        | CL6'000045'                                      |
```

The INDEX macro instruction points to the CAMLST macro instruction. BLDG, the first operand of CAMLST, specifies that a generation index be built. DLEVEL, the second operand, specifies the main storage location of an area into which you have placed the fully qualified name of a generation index. VOLNUM, the third operand, specifies the main storage location of an area into which you have placed the 6-byte serial number of the volume on which the generation index is to be built. DELETE, the fifth operand, specifies that all data sets dropped from the generation data group are to be deleted. The final operand, 4, specifies the number of data sets that are to be maintained in the generation data group.

HOW TO DELETE AN INDEX

You can delete any number of index levels from an existing index structure. Each level of the index is deleted separately. Generation indexes are also removed this way. (You can also delete index levels automatically when they are no longer needed. See "How to Remove Data Set References from the Catalog" in this chapter for details.) You delete each level of the index by using the INDEX and CAMLST macro instructions.

If an index level either has an alias, or has other index levels or data sets cataloged under it, it cannot be deleted.

Example: In the following example, index level C is deleted from index structure A.B.C. The search for the index level starts on the system residence volume.

```
| Name     | Operation | Operand                                          |
|          | INDEX     | DELETE                DELETE INDEX LEVEL C FROM  |
|      Check Exceptional Returns                  INDEX STRUCTURE A.B.C   |
| DELETE   | CAMLST    | DLTX,LEVELC                                      |
| LEVELC   | DC        | CL6'A.B.C'            ONE BLANK FOR DELIMITER    |
```

The INDEX macro instruction points to the CAMLST macro instruction. DLTX, the first operand of CAMLST, specifies that an index level be deleted. LEVELC, the second operand, specifies the main storage location of an area into which you have placed the fully qualified name of the index structure whose lowest level is to be deleted.

HOW TO ASSIGN AN ALIAS

You assign an alias to an index level by using the INDEX and CAMLST macro instructions. An alias can be assigned only to a high level index; e.g., index A of index structure A.B.C can have an alias, but index B cannot. Assigning an alias to a high level index effectively provides aliases for all data sets cataloged under that index. An alias cannot be assigned to a generation index with only one level.

Example: In the following example, index level A is assigned an alias of X. The search for the index level starts on the system residence volume.

```
┌─────────┬─────────┬──────────────────────────────────────────────┐
│ Name    │ Operation│ Operand                                     │
├─────────┼─────────┼──────────────────────────────────────────────┤
│         │ INDEX   │ ALIAS                  BUILD AN ALIAS FOR A HIGH│
│     Check Exceptional Returns            LEVEL INDEX            │
│ ALIAS   │ CAMLST  │ BLDA,DSNAME,,DSALIAS                        │
│ DSNAME  │ DC      │ CL8'A'                 MUST BE 8-BYTE FIELDS │
│ DSALIAS │ DC      │ CL8'X'                                      │
└─────────┴─────────┴──────────────────────────────────────────────┘
```

The INDEX macro instruction points to the CAMLST macro instruction. BLDA, the first operand of CAMLST, specifies that an alias be built. DSNAME, the second operand, specifies the main storage location of an 8-byte area into which you have placed the name of the high level index to be assigned an alias. DSALIAS, the fourth operand, specifies the main storage location of an 8-byte area into which you have placed the alias to be assigned.

HOW TO DELETE AN ALIAS

You delete an alias previously assigned to a high level index by using the INDEX and CAMLST macro instructions.

Example: In the following example, alias X, previously assigned as an alias for index level A, is deleted. The search for the alias starts on the system residence volume.

```
┌─────────┬─────────┬──────────────────────────────────────────────┐
│ Name    │ Operation│ Operand                                     │
├─────────┼─────────┼──────────────────────────────────────────────┤
│         │ INDEX   │ DELALIAS               DELETE AN ALIAS FOR A │
│     Check Exceptional Returns            HIGH LEVEL INDEX       │
│ DELALIAS│ CAMLST  │ DLTA,ALIAS                                  │
│ ALIAS   │ DC      │ CL8'X'                 MUST BE 8-BYTE FIELD  │
└─────────┴─────────┴──────────────────────────────────────────────┘
```

The INDEX macro instruction points to the CAMLST macro instruction. DLTA, the first operand of CAMLST, specifies that an alias be deleted. ALIAS, the second operand, specifies the main storage location of an 8-byte area into which you have placed the alias to be deleted.

HOW TO CONNECT CONTROL VOLUMES

You connect two control volumes by using the INDEX and CAMLST macro instructions. If a control volume is to be connected to the system residence volume, you need supply only the serial number of the volume to be connected and the name of a high level index associated with the volume to be connected.

If a control volume is to be connected to a control volume other than the system residence volume, you must supply the serial numbers of both volumes and the name of a high level index associated with the volume to be connected.

The result of connecting control volumes is that the volume serial number of the control volume connected and the name of a high level index are entered into the volume index of the volume to which it was connected. This entry is called a control volume pointer.

Example: In the following example, the control volume whose serial number is 001555 is connected to the control volume numbered 000155. The name of the high level index is HIGHINDX.

```
┌─────────┬─────────┬──────────────────────────────────────────────┐
│ Name    │ Operation│ Operand                                     │
├─────────┼─────────┼──────────────────────────────────────────────┤
│         │ INDEX   │ CONNECT                      CONNECT TWO CON-│
│     Check Exceptional Returns                  TROL VOLUMES WHOSE│
│ CONNECT │ CAMLST  │ LNKX,INDXNAME,OLDCVOL,NEWCVOL SERIAL NUMBERS ARE│
│ INDXNAME│ DC      │ CL8'HIGHINDX'                000155 AND 001555.│
│ OLDCVOL │ DC      │ CL6'000155'                                  │
│ NEWCVOL │ DC      │ X'30002001'                  2311 DISK STORAGE│
│         │ DC      │ CL6'001555'                                  │
└─────────┴─────────┴──────────────────────────────────────────────┘
```

The INDEX macro instruction points to the CAMLST macro instruction. LNKX, the first operand of CAMLST, specifies that control volumes be connected. INDXNAME, the second operand, specifies the main storage location of an 8-byte area into which you have placed the name of the high level index of the volume to be connected. OLDCVOL, the third operand, specifies the main storage location of a 6-byte area into which you have placed the serial number of the volume to which you are connecting. NEWCVOL, the fourth operand, specifies the main storage location of a 10-byte area into which you have placed the 4-byte binary device code of the volume to be connected followed by the 6-byte area to contain the volume serial number of the volume to be connected.

HOW TO DISCONNECT CONTROL VOLUMES

You disconnect two control volumes by using the INDEX and CAMLST macro instructions. If a control volume is to be disconnected from the system residence volume, you need supply only the name of the high level index associated with the volume to be disconnected.

If a control volume is to be disconnected from a control volume other than the system residence volume, you must supply, in addition to the name of the high level index, the serial number of the control volume from which you want to disconnect.

The result of disconnecting control volumes is that the control volume pointer is removed from the volume index of the volume from which you are disconnecting.

Example: In the following example, the control volume that contains the high level index HIGHINDX is disconnected from the system residence volume.

```
┌─────────┬─────────┬──────────────────────────────────────────────┐
│ Name    │ Operation│ Operand                                     │
├─────────┼─────────┼──────────────────────────────────────────────┤
│         │ INDEX   │ DISCNECT       DISCONNECT TWO CONTROL VOLUMES│
│      Check Exceptional Returns                                  │
│ DISCNECT│ CAMLST  │ DRPX,INDXNAME                               │
│ INDXNAME│ DC      │ CL8'HIGHINDX'  MUST BE 8-BYTE FIELD         │
└─────────┴─────────┴──────────────────────────────────────────────┘
```

The INDEX macro instruction points to the CAMLST macro instruction. DRPX, the first operand of CAMLST, specifies that control volumes be disconnected. INDEXNAME, the second operand, specifies the main storage location of an 8-byte area into which you have placed the name of the high level index of the control volume to be disconnected.

HOW TO CATALOG A DATA SET

You catalog a data set by using the CATALOG and CAMLST macro instructions. The CATALOG macro instruction points to the CAMLST macro instruction; parameters of the CAMLST macro instruction specify the options for cataloging a data set. When the CAT parameter is used, all index levels required to catalog the data set must exist in the catalog. The index structure need not exist when the CATBX parameter is used; any missing index levels are automatically created.

You must build a complete volume list in main storage. This volume list consists of volume pointers for all volumes on which the data set is stored. The first two bytes of the list indicate the number of volume pointers that follow. Each 12-byte volume pointer consists of a 4-byte device code, a 6-byte volume serial number, and a 2-byte data set sequence number. The sequence number is always zero for direct access volumes. (Device codes are contained in Appendix B of this chapter.)

-When Index Levels Exist

When the index levels already exist for a data set, you can use the CAT parameter of the CAMLST macro instruction to catalog the data set. Missing index levels cause an exceptional return code to be set.

Example: In the following example, the data set named A.B.C is cataloged under an existing index structure A.B. The data set is stored on two volumes.

```
┌─────────┬─────────┬──────────────────────────────────────────────┐
│ Name    │ Operation│ Operand                                     │
├─────────┼─────────┼──────────────────────────────────────────────┤
│         │ CATALOG │ ADDABC         CATALOG DATA SET A.B.C. THE  │
│      Check Exceptional Returns       INDEX STRUCTURE A.B. EXISTS│
│ ADDABC  │ CAMLST  │ CAT,DSNAME,,VOLUMES                         │
│ DSNAME  │ DC      │ CL6'A.B.C'     ONE BLANK FOR DELIMITER      │
│ VOLUMES │ DC      │ H'2'           TWO VOLUMES                  │
│         │ DC      │ X'30002001'    2311 DISK STORAGE            │
│         │ DC      │ CL6'000014'    VOLUME SERIAL NUMBER         │
│         │ DC      │ H'0'           DATA SET SEQUENCE NUMBER     │
│         │ DC      │ X'30002001'    2311 DISK STORAGE            │
│         │ DC      │ CL6'000015'    VOLUME SERIAL NUMBER         │
│         │ DC      │ H'0'           SEQUENCE NUMBER              │
└─────────┴─────────┴──────────────────────────────────────────────┘
```

The CATALOG macro instruction points to the CAMLST macro instruction. CAT, the first operand of CAMLST, specifies that a data set be cataloged. DSNAME, the second operand, specifies the main storage location of an area into which you have placed the fully qualified name of the data set to be cataloged. VOLUMES, the fourth operand, specifies the main storage location of the volume list you have built.

-By Creating Required Index Levels

When index levels are missing, you can use the CATBX parameter of the CAMLST macro instruction to automatically create them before cataloging the data set.

Example: In the following example, the index structure A.B is created and data set A.B.C is cataloged. The data set is stored on one volume.

```
| Name     | Operation| Operand                                          |
|          | CATALOG  | CTBXABC          CATALOG DATA SET A.B.C          |
|      Check Exceptional Returns      CREATE NEEDED INDEX LEVELS        |
| CTBXABC  | CAMLST   | CATBX,DSNAME,VOLUMES,DSCBTTR=TTR                 |
| DSNAME   | DC       | CL6'A.B.C'       ONE BLANK FOR DELIMITER         |
| VOLUMES  | DC       | H'1'             ONE VOLUME                      |
|          | DC       | X'30002001'      2311 DISK STORAGE               |
|          | DC       | CL6'000015'      VOLUME SERIAL NUMBER            |
|          | DC       | H'0'             DATA SET SEQUENCE NUMBER        |
| TTR      | DC       | XL3'000103'      TTR OF DSCB IN VTOC             |
```

The CATALOG macro instruction points to the CAMLST macro instruction. CATBX, the first operand of CAMLST, specifies that a data set is to be cataloged and any required higher level indexes are to be created. DSNAME, the second operand, specifies the main storage location of an area into which you have placed the fully qualified name of the data set to be cataloged. VOLUMES, the third operand, specifies the main storage location of the volume list you have built. DSCBTTR=TTR, the fourth operand, specifies the main storage location into which you have placed the relative track address of the DSCB for the data set to be cataloged. The DSCBTTR operand is optional and is ignored for data sets residing on more than one volume.

HOW TO REMOVE DATA SET REFERENCES FROM THE CATALOG

You remove data set references from the catalog by using the CATALOG and CAMLST macro instructions. Two options are available: simply remove references, or remove references and delete any indexes that are no longer needed.

-Uncatalog and Retain Index Levels

When the UNCAT operand of the CAMLST macro instruction is used, a data set reference is removed, but all index levels are retained.

Example: In the following example, references to data set A.B.C are removed from the catalog.

```
┌─────────┬─────────┬──────────────────────────────────────────────┐
| Name    | Operation| Operand                                     |
├─────────┼─────────┼──────────────────────────────────────────────┤
|         | CATALOG | REMOVE              REMOVE REFERENCES TO DATA |
|     Check Exceptional Returns            SET A.B.C FROM THE CATALOG |
| REMOVE  | CAMLST  | UNCAT,DSNAME                                 |
| DSNAME  | DC      | CL6'A.B.C'          ONE BLANK FOR DELIMITER  |
└─────────┴─────────┴──────────────────────────────────────────────┘
```

The CATALOG macro instruction points to the CAMLST macro instruction. UNCAT, the first operand of CAMLST, specifies that references to a data set be removed from the catalog. DSNAME, the second operand, specifies the main storage location of an area into which you have placed the fully qualified name of the data set whose references are to be removed.

-Uncatalog and Remove Index Levels

When the UNCATDX operand of the CAMLST macro instruction is used, a data set reference and unneeded indexes, with the exception of the highest-level index, are removed from the catalog.

Example: In the following example, references to data set A.B.C are removed from the catalog. Index B is removed unless it contains references to other data sets. Index A remains because it is the highest level index.

```
┌─────────┬─────────┬──────────────────────────────────────────────┐
| Name    | Operation| Operand                                     |
├─────────┼─────────┼──────────────────────────────────────────────┤
|         | CATALOG | RMDSNNDX       REMOVE REFERENCES TO DATA     |
|     Check Exceptional Returns          SET A.B.C FROM THE CATALOG |
| RMDSNNDX| CAMLST  | UCATDX,DSNAME  AND REMOVE INDEXES            |
| DSNAME  | DC      | CL6'A.B.C'     ONE BLANK FOR DELIMITER       |
└─────────┴─────────┴──────────────────────────────────────────────┘
```

The CATALOG macro instruction points to the CAMLST macro instruction. UNCATDX, the first operand, specifies that references to a data set be removed from the catalog. DSNAME, the second operand, specifies the main storage location of an area into which you have placed the fully qualified name of the data set whose references are to be removed.

HOW TO RECATALOG A DATA SET

You recatalog a cataloged data set by using the CATALOG and CAMLST macro instructions. Recataloging is usually performed when new volume pointers must be added to the volume list of a data set.

You must build a complete volume list in main storage. This volume list consists of volume pointers for all volumes on which the data set is stored. The first two bytes of the list indicate the number of volume pointers that follow. Each 12-byte volume pointer consists of a 4-byte device code, a 6-byte volume serial number, and a 2-byte data set sequence number. The sequence number is always zero for direct access volumes. (Device codes are contained in Appendix B of this chapter.)

Example: In the following example, the data set named A.B.C is recataloged. A new volume pointer is added to the volume list, which previously contained only two volume pointers.

```
| Name     | Operation | Operand                                          |
|----------|-----------|--------------------------------------------------|
|          | CATALOG   | RECATLG                  RECATALOG DATA SET A.B.C,|
|     Check Exceptional Returns       ADDING A NEW VOLUME          |
|          |           |                          POINTER TO THE VOLUME   |
| RECATLG  | CAMLST    | RECAT,DSNAME,,VOLUMES    LIST.                   |
| DSNAME   | DC        | CL6'A.B.C'               ONE BLANK FOR DELIMITER |
| VOLUMES  | DC        | H'3'                     THREE VOLUMES           |
|          | DC        | X'30002001'              2311 DISK STORAGE       |
|          | DC        | CL6'000014'              VOLUME SERIAL NUMBER    |
|          | DC        | H'0'                     SEQUENCE NUMBER         |
|          | DC        | X'30002001'              2311 DISK STORAGE       |
|          | DC        | CL6'000015'              VOLUME SERIAL NUMBER    |
|          | DC        | H'0'                     SEQUENCE NUMBER         |
|          | DC        | X'30002001'              2311 DISK STORAGE       |
|          | DC        | CL6'000016'              VOLUME SERIAL NUMBER    |
|          | DC        | H'0'                     SEQUENCE NUMBER         |
```

The CATALOG macro instruction points to the CAMLST macro instruction. RECAT, the first operand of CAMLST, specifies that a data set be recataloged. DSNAME, the second operand, specifies the main storage location of an area into which you have placed the fully qualified name of the data set to be recataloged. VOLUMES, the fourth operand, specifies the main storage location of the volume list you have built.

HOW TO READ A DATA SET CONTROL BLOCK FROM THE VOLUME TABLE OF CONTENTS

You can read a data set control block (DSCB) into main storage by using the OBTAIN and CAMLST macro instructions. There are two ways to specify the DSCB that you want read: by using the name of the data set associated with the DSCB, or by using the absolute track address of the DSCB.

When you specify the name of the data set, a format 1 (identifier) DSCB is read into main storage. To read a DSCB other than a format 1 DSCB, you must specify an absolute track address. (DSCB formats and field descriptions are contained in the System Control Block publication).

When a data set name is specified, the 96-byte data portion of the format 1 DSCB, and the absolute track address of the DSCB are read into main storage. When the absolute track address of a DSCB is specified, the 44-byte key portion and the 96-byte data portion of the DSCB are read into main storage.

Example: In the following example, the format 1 DSCB for data set A.B.C is read into main storage. The serial number of the volume containing the DSCB is 770655.

```
| Name     | Operation | Operand                                          |
|----------|-----------|--------------------------------------------------|
|          | OBTAIN    | DSCBABC                  READ DSCB FOR DATA      |
|     Check Exceptional Returns       SET A.B.C INTO MAIN|
| DSCBABC  | CAMLST    | SEARCH,DSABC,VOLNUM,WORKAREA STORAGE AREA NAMED  |
| DSABC    | DC        | CL44'A.B.C'              WORKAREA.  96-BYTE      |
| VOLNUM   | DC        | CL6'770655'              DATA PORTION IS         |
| WORKAREA | DS        | 0D                       READ.  THE REST OF      |
|          | DS        | 148C                     THE AREA IS USED BY|
|          |           |                          THE OBTAIN ROUTINE      |
```

The OBTAIN macro instruction points to the CAMLST macro instruction. SEARCH, the first operand of CAMLST, specifies that a DSCB be read into main storage. DSABC, the second operand, specifies the main storage location of a 44-byte area into which you have placed the fully qualified name of the data set whose associated DSCB is to be read. VOLNUM, the third operand, specifies the main storage location of a 6-byte area into which you have placed the serial number of the volume containing the required DSCB. WORKAREA, the fourth operand, specifies the main storage location of a 148-byte work area that is to contain the DSCB.

After execution of these macro instructions, the first 96 bytes of the work area contain the data portion of the format 1 DSCB; the next five bytes contain the absolute track address of the DSCB.

HOW TO DELETE A DATA SET

You delete a data set stored on direct access volumes by using the SCRATCH and CAMLST macro instructions. This causes all data set control blocks (DSCB) for the data set to be deleted, and all space occupied by the data set to be made available for reallocation. If the data set to be deleted is sharing a split cylinder, the space will not be made available for reallocation until all data sets on the split cylinder are deleted.

A data set cannot be deleted if the expiration date in the format 1 (identifier) DSCB has not passed, unless you choose to ignore the expiration date. You can ignore the expiration date by using the OVRD option in the CAMLST macro instruction.

If a data set to be deleted is stored on more than one volume, either a device must be available on which to mount the volumes, or at least one volume must be mounted. In addition, all other required volumes must be serially mountable. Certain volumes, such as the system residence volume, must always be mounted.

When deleting a data set, you must build a complete volume list in main storage. This volume list consists of volume pointers for all volumes on which the data set is stored. The first two bytes of the list indicate the number of volume pointers that follow. Each 12-byte volume pointer consists of a 4-byte device code, a 6-byte volume serial number, and a 2-byte scratch status code. (Device codes are contained in Appendix B of this chapter.)

Volumes are processed in the order that they appear in the volume list. Those volumes that are pointed to at the beginning of the list are processed first. If a volume is not mounted, a message is issued to the operator requesting him to mount the volume. This is done if you indicate the I/O device on which unmoumted volumes are to be mounted by loading register zero with the address of the UCB associated with the device to be used. If you do not load register zero with the UCB address, its contents must be zero, and at least one volume in the volume list must be mounted before the SCRATCH macro instruction is executed.

If the operator cannot mount the requested volume, he issues a reply indicating that he cannot fulfill the request. A condition code is then set in the last byte of the volume pointer (the second byte of the scratch status code) for the unavailable volume, and the next volume indicated in the volume list is processed or requested.

Example: In the following example, data set A.B.C is deleted from two volumes. The expiration date in the format 1 DSCB is ignored.

```
| Name     | Operation| Operand                                          |
|----------|----------|--------------------------------------------------|
|          | SR       | 0,0                              SET REG 0 TO ZERO   |
|          | SCRATCH  | DELABC                           DELETE DATA SET     |
|     Check Exceptional Returns                       A.B.C. FROM TWO  |
| DELABC   | CAMLST   | SCRATCH,DSABC,,VOLIST,,OVRD      VOLUMES, IGNORING   |
| DSABC    | DC       | CL44'A.B.C'                      THE EXPIRATION      |
| VOLIST   | DC       | H'2'                             DATE IN THE DSCB.   |
|          | DC       | X'30002001'                      2311 DISK STORAGE   |
|          | DC       | CL6'000017'                      VOLUME SERIAL NO.   |
|          | DC       | H'0'                             SCRATCH STATUS CODE |
|          | DC       | X'30002001'                      2311 DISK STORAGE   |
|          | DC       | CL6'000018'                      VOLUME SERIAL NO.   |
|          | DC       | H'0'                             SCRATCH STATUS CODE |
```

The SCRATCH macro instruction points to the CAMLST macro instruction. SCRATCH, the first operand of CAMLST, specifies that a data set be deleted. DSABC, the second operand, specifies the main storage location of a 44-byte area into which you have placed the fully qualified name of the data set to be deleted. VOLIST, the fourth operand, specifies the main storage location of the volume list you have built. OVRD, the sixth operand, specifies that the expiration date be ignored in the DSCB of the data set to be deleted.

HOW TO RENAME A DATA SET

You rename a data set stored on direct access volumes by using the RENAME and CAMLST macro instructions. This causes the data set name in all identifier (format 1) data set control blocks (DSCB) for the data set to be replaced by the new name that you supply.

If a data set to be renamed is stored on more than one volume, either a device must be available on which to mount the volumes, or at least one volume must be mounted. In addition, all other required volumes must be serially mountable. Certain volumes, such as the system residence volume, must always be mounted.

When renaming a data set, you must build a complete volume list in main storage. This volume list consists of volume pointers for all volumes on which the data set is stored. The first two bytes of the list indicate the number of volume pointers that follow. Each 12-byte volume pointer consists of a 4-byte device code, a 6-byte volume serial number, and a 2-byte rename status code. (Device codes are contained in Appendix B of this chapter.)

Volumes are processed in the order they appear in the volume list. Those volumes that are pointed to at the beginning of the list are processed first. If a volume is not mounted, a message is issued the operator requesting him to mount the volume. This is done if you indicate the I/O device on which unmounted volumes are to be mounted by loading register zero with the address of the UCB associated with the device to be used. If you do not load register zero with the UCB address, its contents must be zero, and at least one volume in the volume list must be mounted before the RENAME macro instruction is executed.

If the operator cannot mount the requested volume, he issues a reply indicating that he cannot fulfill the request. A condition code is then set in the last byte of the volume pointer (the secondary byte of the rename status code) for the unavailable volume, and the next volume indicated in the volume list is processed or requested.

Example: In the following example, data set A.B.C is renamed D.E.F. The data set extends across two volumes.

```
| Name     | Operation | Operand                                  |
|----------|-----------|------------------------------------------|
|          | SR        | 0,0                    SET REG 0 TO ZERO |
|          | RENAME    | DSABC                  CHANGE DATA SET   |
|      Check Exceptional Returns               NAME A.B.C. TO   |
| DSABC    | CAMLST    | RENAME,OLDNAME,NEWNAME,VOLIST D.E.F      |
| OLDNAME  | DC        | CL44'A.B.C'                              |
| NEWNAME  | DC        | CL44'D.E.F'                              |
| VOLIST   | DC        | H'2'                   TWO VOLUMES       |
|          | DC        | X'30002001'            2311 DISK STORAGE |
|          | DC        | CL6'000017'            VOLUME SERIAL NO. |
|          | DC        | H'0'                   RENAME STATUS CODE|
|          | DC        | X'30002001'            2311 DISK STORAGE |
|          | DC        | CL6'000018'            VOLUME SERIAL NO. |
|          | DC        | H'0'                   RENAME STATUS CODE|
```

The RENAME macro instruction points to the CAMLST macro instruction. RENAME, the first operand of CAMLST, specifies that a data set be renamed. OLDNAME, the second operand, specifies the main storage location of a 44-byte area into which you have placed the fully qualified name of the data set to be renamed. NEWNAME, the third operand, specifies the main storage location of a 44-byte area into which you have placed the new name of the data set. VOLIST, the fourth operand, specifies the main storage location of the volume list you have built.

How to Share Space on a Volume Initialized Under DOS

With the addition to the OS DADSM allocation program of a routine to convert a DOS format VTOC to an OS format VTOC, it is now possible to share the space on such a volume (one initialized under DOS) between data sets written by users using DOS and users using OS. The degree and limits of sharing are:

- The OS user may now request space in any standard OS form of space allocation, that is: TRK, CYL, average block size, and ABSTR.
- The OS stand-alone utility program IBCRCVRP does not accept alternate track assignment made under DOS. If the volume has any alternate tracks assigned under DOS, and additional alternate tracks must be assigned, the DOS assign alternate track program must be used to perform that function.

The net effect is that OS and DOS may share a volume, but the data sets written under each system can only be read under the system under which they were written.

Appendix A: Catalog Block Entries

This section describes the contents of all catalog entries.

Control Entries

A volume index control entry is always the first entry in a volume index. The volume index control entry is 22 bytes long and contains eight fields.

Field 1: Name field (8 bytes) -- contains only a binary one to ensure that this entry is the first entry in the first block of the index.

Field 2: Last block address (3 bytes) -- contains the relative track address of the last block in the volume index. The address is in the form TTR.

Field 3: Halfword count (1 byte) -- contains a binary five to indicate that five half words follow.

Field 4: Catalog upper limit (3 bytes) -- contains the relative track address of the last block in the catalog data set. The address is in the form TTR.

Field 5: Zero field (1 byte) -- contains binary zeros.

Field 6: First available block address (3 bytes) -- contains the relative track address of the unused block in the catalog that is closest to the beginning of the catalog data set.

Field 7: Zero field (1 byte) -- contains binary zeros.

Field 8: Unused bytes in last block (2 bytes) -- contains the binary count of the number of unused bytes in the last block of the volume index.

An index control entry is the first entry in all indexes except volume indexes. The index control entry is 18 bytes long and contains six fields.

Field 1: Name field (8 bytes) -- contains only a binary one to ensure that this entry, because it has the lowest binary name value, is the first entry in the first block of the index.

Field 2: Last block address (3 bytes) -- contains the relative track address of the last block assigned to the index. The address is in the form TTR.

Field 3: Halfword count (1 byte) -- contains a binary three to indicate that three half words follow.

Field 4: Index lower limit (3 bytes) -- contains the relative track address of the block in which this entry appears. The address is in the form TTR.

Field 5: Number of aliases (1 byte) -- contains the binary count of the number of aliases assigned to the index. If the index is not a high level index, this field is zero.

Field 6: Unused bytes in last block (2 bytes) -- contains the binary count of the number of unused bytes remaining in the last block of the index.

An index link entry is the last entry in all index blocks. The entry is 12 bytes long and contains three fields.

Field 1: Name field (8 bytes) -- contains only the hexadecimal number FF to ensure that this entry, because it has the highest binary name value, will appear as the last entry in any index block.

Field 2: Link address (3 bytes) -- contains the relative track address of the next block of the same index, if there is a next block in the index. Otherwise, the field contains binary zeros.

Field 3: Halfword count (1 byte) -- contains a binary zero to indicate that no additional fields follow.

Pointer Entries

An index pointer entry can appear in all indexes except generation indexes. The entry is 12 bytes long and contains three fields.

Field 1: Name field (8 bytes) -- contains the name of the index being pointed to by field 2.

Field 2: Index address (3 bytes) -- contains the relative track address of the first block of the index named in field 1. The address is in the form TTR.

Field 3: Halfword count (1 byte) -- contains a binary zero to indicate that no additional fields follow.

A data set pointer entry can appear in any index. It contains the simple name of a data set and from one to five 12-byte fields that each identify a volume on which the named data set resides. If the data set resides on more than five volumes, a volume control block must be used to point to the volumes. The volume control block is identified by a volume control block pointer entry, not a data set pointer entry.

The data set pointer entry varies in length. The length is determined by the formula (14+12m), where m is the number of volumes containing the data set. The variable m can be from 1 through 5. The data set pointer entry can appear in any index, and it contains five fields.

Field 1: Name field (8 bytes) -- contains the simple name of the data set whose volumes are identified in field 5.

Field 2: DSCB TTR (3 bytes) -- contains the track address (TTR) of the data set control block if the data set resides on only one volume. If the data set resides on more than one volume, this field contains a binary zero.

Field 3: Halfword count (1 byte) -- contains the binary count of the number of half words that follow. The number is found by the formula (6m+1), where m is the number of volumes on which the data set resides. The variable m can be from 1 through 5.

Field 4: Volume count (2 bytes) -- contains the binary count of the number of volumes identified in field 5 of this entry.

Field 5: Volume entries (12 to 60 bytes) -- contains from one to five 12-byte entries, each of which identifies a volume on which the data set resides. Each entry contains a 4-byte device code, a 6-byte volume serial number, and a 2-byte data set sequence number. The data set sequence number is zero for direct access volumes.

A volume control block pointer entry can appear in any index. It can identify up to 20 volumes. The entry is 14 bytes long and contains four fields.

Field 1: Name field (8 bytes) -- contains the last name of the qualified name of the data set identified by this entry. The data set resides on the volumes whose serial numbers are given in the volume control block pointed to by field 2.

Field 2: Address field (3 bytes) -- contains the relative track address of the volume control block identifying the volumes containing the data set named in field 1. The address is in the form TTR.

Field 3: Halfword count (1 byte) -- contains a binary one to indicate that one half word follows.

Field 4: Zero field (2 bytes) -- contains binary zeros.

A control volume pointer entry can appear only in volume indexes. It is 18 bytes long and contains four fields.

Field 1: Name field (8 bytes) -- contains a high level index name that appears in the volume index of the control volume identified in field 4.

Field 2: Address field (3 bytes) -- contains binary zeros.

Field 3: Halfword count (1 byte) -- contains a binary three to indicate that three half words follow.

Field 4: Control volume serial number (6 bytes) -- contains the serial number of the control volume whose volume index contains an entry identifying the high level index name in field 1.

A new control volume pointer entry can appear only in volume indexes. It is 22 bytes long and contains 5 fields.

Field 1: Name field (8 bytes) contains a high level index name that appears in the volume index of the control volume identified in fields 4 and 5.

Field 2: Address field (3 bytes) contains binary zeros.

Field 3: Halfword count (1 byte) contains a binary 5 to indicate that five halfwords follow.

Field 4: Control volume device code (4 bytes) contains the 4-byte binary device code of the control volume whose index contains an entry identifying the high level index name in field 1.

Field 5: Control volume serial number (6 bytes) contains the serial number of the control volume whose index contains an entry identifying the high level index name in field 1.

An alias entry can appear in volume indexes only. An alias entry is 20 bytes long and contains four fields.

Field 1: Name field (8 bytes) -- contains the alias of the high level index identified in field 2.

Field 2: Address field (3 bytes) -- contains the relative track address of the first block of the index named in field 4. The address is in the form TTR.

Field 3: Halfword count (1 byte) -- contains a binary four to indicate that four half words follow.

Field 4: True name field (8 bytes) -- contains the name of the index whose alias appears in field 1. The address of the index is in field 2.

A generation index pointer entry can appear in all indexes except generation indexes. The entry is 16 bytes long and contains six fields.

Field 1: Name field (8 bytes) -- contains the name of the generation index whose address is contained in field 2.

Field 2: Address field (3 bytes) -- contains the relative track address of the generation index named in field 1. The address is in the form TTR.

Field 3: Halfword count (1 byte) -- contains a binary two to indicate that two half words follow.

Field 4: Flags (1 byte) -- contains flags that govern the uncataloging of data sets as specified by the DELETE and EMPTY options of the INDEX macro instruction. The options and their hexadecimal codes are as follows:

EMPTY=01 DELETE=02 EMPTY and DELETE=03

Field 5: Maximum generations allowed (1 byte) -- contains the binary count of the maximum number of generations allowed in the index at one time as specified in the INDEX macro instruction.

Field 6: Current generation count (2 bytes) -- contains the binary count of the number of generations cataloged in the index.

The Volume Control Block Contents

A volume control block is composed of one or more volume-list blocks. Each volume-list block contains an 8-byte key and a 256-byte data portion. The data portion of the volume-list block can identify up to 20 volumes on which a data set is recorded. The format of the volume list block is as follows:

Field 1: Number of volumes (2 bytes) -- the first volume-list block contains the binary count of volumes on which the data set is stored; the value of this field is reduced by 20 for each subsequent volume-list block. If a data set is on 61 volumes, for example, it has four volume-list blocks. The first field of each block contains 61,41,21, and 1, respectively.

Field 2: Volume identification (12 to 240 bytes) -- contains from 1 to 20 12-byte entries, each of which identifies a volume on which the data set resides. Each entry contains a 4-byte device code, a 6-byte volume serial number, and a 2-byte data set sequence number. The data set sequence number is zero for direct access volumes.

Field 3: Zero field (10 bytes) -- contains binary zeros.

Field 4: Chain address (3 bytes) -- contains the relative track address of the next block of this volume control block, if additional blocks exist. The address is in the form TTR. If this is the last block of the volume control block, the field contains a binary zero. If this field is not zero, this block must contain twenty 12-byte fields identifying volumes of the data set.

Field 5: Zero field (1 byte) -- contains binary zeros.

Appendix B: Device Code Designations

Device	Features	Device Code Designation (In Hexadecimal)
IBM 2400 Series Magnetic Tape Units		30008001
IBM 2400 Series Magnetic Tape Units	7-track Compatibility	30808001
IBM 2400 Series Magnetic Tape Units	7-track Compatibility Data Conversion	30C08001
IBM 2400 Series Magnetic Tape Units	Phase Encoding	34008001
IBM 2400 Series Magnetic Tape Units	Phase Encoding with Dual Density	34208001
IBM 2311 Disk Storage Drive		30002001
IBM 2301 Drum Storage		30402002
IBM 2302 Disk Storage		30002004
IBM 2303 Drum Storage		30002003
IBM 2314 Direct Access Storage Facility		30C02008
IBM 2321 Data Cell		30002005
IBM 2305 Fixed Head Storage Model 1		30002006
IBM 2305 Fixed Head Storage Model 2		30002007
IBM 3330 Disk Storage		30502009
IBM 3400 Series Magnetic Tape Units	Phase Encoding	34008003
IBM 3400 Series Magnetic Tape Units	Phase Encoding with Dual Density	34208003
IBM 3400 Series Magnetic Tape Units	7-track	34C08003

Note: These and other device codes are also enumerated under the DEVTYPE macro instruction in the chapter: "System Macro Instructions."

IECDSECT, IEFUCBOB, and IEFJFCBN Macro Instructions

If you want to use the IECDSECT, IEFJFCBN, and IEFUCBOB macro instructions, you must either add these macro definitions to the macro library (SYS1.MACLIB) or place them in a separate partitioned data set and concatenate this data set to the macro library. Expansions of these macros appear in the microfiche for some open/close/EOV modules.

This chapter contains the following:

- The format of the macro instructions.
- The job control and utility statements needed to add the macro instructions to the library.

Information about label handling routines may be found in the publication IBM System/360 Operating System: Tape Labels, GC28-6680.

IECDSECT Macro Instruction

This macro instruction defines the s-mbolic names of fields in the work area used by the OPEN, CLOSE, TCLOSE, and EOV routines. Consult IBM System/360 Operating System: Input/Output Support (OPEN/CLOSE/EOV) Program Logic Manual, GY28-6609, for a description of fields in the work area. Code this macro instruction with blank name and operand fields, and precede it with a DSECT statement. Note: The IEFJFCBN macro instruction is used in the assembly of IECDSECT. The macro definition for IEFJFCBN must be present in the macro-library (SYS1.MACLIB) for successful definition of all fields in the work area.

Name	Operation	Operand
	IECDSECT	

Control Statements Required

```
//jobname       JOB     {parameters}
//stepname      EXEC    PGM=IEBUPDTE,PARM=NEW
//SYSPRINT      DD      SYSOUT=A
//SYSUT2        DD      DSNAME=SYS1.MACLIB,DISP=OLD
//SYSIN         DD      DATA
./        ADD   NAME=IECDSECT,LIST=ALL
                .
                .
                .
                IECDSECT Macro definition
                .
                .
                .
./        ENDUP
/*
```

IEFUCBOB Macro Instruction

This macro instruction defines the symbolic names of all fields in the unit control block (UCB). Code this macro instruction with blank name and operand fields, and precede it with a DSECT statement.

Name	Operation	Operand
	IEFUCBOB	

Control Statements Required

```
//jobname       JOB     {parameters}
//stepname      EXEC    PGM=IEBUPDTE,PARM=NEW
//SYSPRINT      DD      SYSOUT=A
//SYSUT2        DD      DSNAME=SYS1.MACLIB,DISP=OLD
//SYSIN         DD      DATA
./        ADD   NAME=IEFUCBOB,LIST=ALL
                .
                .
                .
                IEFUCBOB Macro definition
                .
                .
                .
./        ENDUP
/*
```

The IEFUCBOB macro definition may be found in the SYS1.MODGEN data set on one of the system generation DLIB disks. SYS1.MODGEN may be concatenated to SYS1.MACLIB during your assembly.

IEFJFCBN Macro Instruction

This macro instruction defines the symbolic names of all fields in the job file control block (JFCB). Code this macro instruction with blank name and operand fields, and precede it with a DSECT statement.

Name	Operation	Operand
	IEFJFCBN	

Control Statements Required

```
//jobname       JOB     (parameters)
//stepname      EXEC    PGM=IEBUPDTE,PARM=NEW
//SYSPRINT      DD      SYSOUT=A
//SYSUT2        DD      DSNAME=SYS1.MACLIB,DISP=OLD
//SYSIN         DD      DATA
./        ADD   NAME=IEFJFCBN,LIST=ALL
                .
                .
                .
                IEFJFCBN macro definition
                .
                .
                .
./        ENDUP
/*
```

Execute Channel Program (EXCP) Macro Instruction

This chapter contains a general description of the function and application of the execute channel program (EXCP) macro instruction, accompanied by descriptions of specific control blocks and macro instructions used with EXCP. Factors that affect the operation of EXCP, such as device variations and program modification, are also discussed.

The EXCP macro instruction provides you with a device-dependent means of performing the I/O operations. Before reading this chapter, you should be familiar with system functions and with the structure of control blocks, as well as with the operational characteristics of the I/O devices required by your channel programs. Operational characteristics of specific I/O devices are contained in IBM System Reference Library publications for each device.

PREREQUISITE PUBLICATIONS

The IBM System/360 Operating System: Supervisor Services and Macro Instructions publication (GC28-6646) explains the standard procedures for I/O processing under the operating system.

The IBM System/360 Operating System: Assembler Language publication (GC28-6514) contains the information necessary to code programs in the assembler language.

The IBM System/360 Operating System: Data Management Macro Instructions publication (GC26-3794) describes the system macro instructions that can be used in programs coded in the assembler language.

The IBM System/360 Operating System: System Control Blocks publication (GC28-6628) contains format and field descriptions of the system control blocks referred to in this chapter.

Execute Channel Program (EXCP) Macro Instruction

Execute channel program (EXCP) is a macro instruction of System/360 Operating System that causes a supervisor-call interruption to pass control to the input/output supervisor. EXCP also provides the input/output supervisor with control information regarding a channel program to be executed. When the IBM standard data access methods are being used, the access method routines are responsible for issuing EXCP. If you are not using the standard access methods, you may issue EXCP directly. Direct use of EXCP provides you with device dependence in organizing data and controlling I/O devices.

You issue EXCP primarily for I/O programming situations to which the standard access methods do not apply. When you are writing your own data access methods, you must include EXCP for I/O operations. EXCP must also be used for processing of nonstandard labels, including the reading and writing of labels and the positioning of magnetic tape volumes.

To issue EXCP, you must provide a channel program (a list of channel command words) and several control blocks in your program area. The input/output supervisor then schedules I/O requests for the device you have specified, executes the specified I/O commands, handles I/O interruptions, directs error recovery procedures, and posts the results of the I/O requests.

When planning EXCP operations and appendages for use on central processing units with parallel processing, special precautions must be observed. Examples of such central processing units are the IBM System/360 Models 91 and 195 and the IBM System/370 Model 195 that can execute instructions in a sequence other than the physical sequence in which they appear in a listing. Such a central processing unit maintains logical consistency in its own operations, including the beginning and ending of I/O operations. However, it is impossible for such a central processing unit to maintain consistency with operations performed by asynchronous units. This type of central processing unit recognizes a special "no operation" to force sequential operations in the environments where it might be required. The appropriate hardware manual should be carefully studied before coding EXCP and appendage routines for this type of central processing unit.

Refer to the topic "Block Multiplexer Channel Programming Notes" for special conditions encountered with command retry.

Use of EXCP In System and Problem Programs

This section briefly explains the procedures performed by the system and the programmer when the EXCP macro instruction is issued by the routines of the standard data access methods. The additional procedures that you must perform when issuing the EXCP macro instruction yourself are then described by direct comparison.

SYSTEM USE OF EXCP

When using a standard data access method to perform I/O operations, the programmer is relieved of coding channel programs, and of constructing the control blocks necessary for the execution of channel programs. To permit I/O operations to be handled by an access method, the programmer need only issue the following macro instructions:

- A DCB macro instruction that produces a data control block (DCB) for the data set to be retrieved or stored. If appendages are not being used, a short DCB is constructed. Such a DCB does not support reduced error recovery.
- An OPEN macro instruction that initializes the data control block and produces a data extent block (DEB) for the data set.
- A macro instruction (e.g., GET, WRITE) that requests I/O operations.

Access method routines will then:

1. Create a channel program that contains channel commands for the I/O operations on the appropriate device.
2. Construct an input/output block (IOB) that contains information about the channel program.
3. Construct an event control block (ECB) that is later supplied with a completion code each time the channel program terminates.
4. Issue an EXCP macro instruction to pass the address of the IOB to the routines that initiate and supervise the I/O operations.

The input/output supervisor will then:

5. Schedule the I/O request.
6. Issue a start input/output (SIO) instruction to activate the I/O device.
7. Process I/O interruptions and schedule error recovery procedures, when necessary.
8. Place a completion code in the event control block after the channel program has been executed.

The programmer is not concerned with these procedures and does not know the status of I/O operations until they are completed. Device-dependent operations are limited to those provided by the macro instructions of the particular access method selected.

PROGRAMMER USE OF EXCP

If you wish to issue the EXCP macro instruction directly, you must perform the procedures that the access methods perform, as summarized in items 1 through 4 of the preceding discussion. You must, in addition to constructing and opening the data control block with the DCB and OPEN macro instructions, construct a channel program, an input/output block, and an event control block before you can issue the EXCP macro instruction. The input/output supervisor always handles items 5 through 8.

After issuing the EXCP macro instruction, you should issue a WAIT macro instruction specifying the event control block to determine whether the channel program has terminated. If volume switching is necessary, you must issue an EOV macro instruction. When processing of the data set has been completed, you should issue a CLOSE macro instruction.

EXCP Requirements

This section describes the channel program that you must provide in order to issue the EXCP macro instruction. The control blocks that you must either construct directly, or cause to be constructed by use of macro instructions, are also described.

CHANNEL PROGRAM

The channel program supplied by you and executed through EXCP is composed of channel command words (CCWs) on doubleword boundaries. Each channel command word specifies a command to be executed and, for commands initiating data transfer, the area to or from which the data is to be transferred. Channel command word formats used with specific I/O devices can be found in IBM Systems Reference Library publications for each device. You should also see the restrictions related to channel commands for direct-access storage devices in the following section, "Initiation of Channel Program." All channel command words described in these publications can be used, with the exception of REWIND and UNLOAD (RUN).

Data and Command Chaining

Chaining is the successive loading of channel command words into a channel from contiguous doubleword locations in main storage. Data chaining occurs when a new channel command word loaded into the channel defines a new storage area for the original I/O operation. Command chaining occurs when the new channel command word specifies a new I/O operation. For detailed information about chaining, refer to the IBM System/360: Principles of Operation publication (GA22-6821).

To specify either data chaining or command chaining, you must set appropriate bits in the channel command word, and indicate the type of chaining in the input/output block. Both data and command chaining should not be specified in the same channel command word; if they are, data chaining takes precedence.

When a channel program includes a list of channel command words that chain data for reading operations, no channel command word may alter the contents of another channel command word in the same list. (If such alteration were allowed, specifications could be placed into a channel command word without being checked for validity. If the specifications were incorrect, the error could not be detected until the chain was completed. Data could be read into incorrect locations and the system could not correct the error.)

CONTROL BLOCKS

When using the EXCP macro instruction, you must be familiar with the function and structure of an input/output block (IOB), an event control block (ECB), a data control block (DCB), and a data extent block (DEB). Brief descriptions of these control blocks follow. Their fields are illustrated in the section "EXCP Programming Specifications."

Input/Output Block (IOB)

The input/output block is used for communication between the problem program and the system. It provides the addresses of other control blocks, and maintains information about the channel program, such as the type of chaining and the progress of I/O operations. You must define the input/output block and specify its address as the only parameter of the EXCP macro instruction.

Event Control Block (ECB)

The event control block provides you with a completion code that describes whether the channel program was completed with or without error. A WAIT macro instruction for synchronizing I/O operations with the problem program must be directed to the event control block. You must define the event control block and specify its address in the input/output block.

Data Control Block (DCB)

The data control block provides the system with information about the characteristics and processing requirements of a data set to be read or written by the channel program. A data control block must be produced by a DCB macro instruction that includes parameters for EXCP. If appendages are not being used, a short DCB is constructed. Such a DCB does not support reduced error recovery. You specify the address of the data control block in the input/output block.

Data Extent Block (DEB)

The data extent block contains one or more extent entries for the associated data set, as well as other control information. An extent defines all or part of the physical boundaries on an I/O device occupied by, or reserved for, a particular data set. Each extent entry contains the address of a unit control block (UCB), which provides information about the type and location of an I/O device. More than one extent entry can contain the same UCB address. (Unit control blocks are set up at system generation time and need not concern you.) For all I/O devices supported by the operating system, the data extent block is produced during execution of the OPEN macro instruction for the data control block. The system places the address of the data extent block into the data control block. (Opening an EXCP data set with DSORG=IS will not produce an ISAM section in the DEB.)

Channel Program Execution

This section explains how the system uses your channel program and control blocks after the EXCP macro instruction has been issued.

INITIATION OF CHANNEL PROGRAM

By issuing the EXCP macro instruction, you request the execution of the channel program specified in the input/output block. The input/output supervisor checks the request for validity by ensuring that the required control blocks contain the correct information. If they do not, abnormal termination procedures are initiated. A program check occurs if the control blocks are not on correct boundaries.

The input/output supervisor obtains the address of the data control block from the input/output block and the address of the data extent block from the data control block. From the data extent block, the system obtains the address of the unit control block (UCB) for the desired I/O device. To protect and facilitate reference to the addresses of the IOB, DEB, and UCB, the input/output supervisor places these addresses, along with other information about the channel program, into an area called a request element. The request element is used by the input/output supervisor for forming queues to keep track of I/O requests. A channel program's request element is "available" if the information it contains is no longer to be used by the input/output supervisor and if it is ready to receive information about another request. When a request element is "made available", it is removed from all request queues and placed on a queue of available request elements.

You are not concerned with the contents of the request element unless you have provided appendage routines, as explained in the section "Appendages."

After completing the request element for the channel program, the input/output supervisor determines whether a channel and the requested I/O device are ready for the channel program. If they are not ready, the request element is placed into the appropriate queue, and control is returned to the problem program. The channel program is subsequently executed when the channel and device are ready.

To initiate execution of the channel program, the system obtains its address from the input/output block, places this address into the channel address word (CAW), and issues a start input/output (SIO) instruction.

Before issuing the SIO instruction for direct access devices, the system issues the initial seek, which is overlapped with other operations. You specify the seek address in the input/output block. When the seek has completed, the system constructs a command chain to reissue the seek, set the file mask specified in the data extent block, and pass control to your channel program. (When using the operating system, you cannot issue the initial seek or set the file mask yourself. The file mask is set to prohibit Seek Cylinder commands, or, if space is allocated by tracks, Seek Track commands. If the data set is opened for INPUT or RDBACK, Write commands are also prohibited.)

Before issuing SIO for magnetic tape devices, the system constructs a command chain to set the mode specified in the data extent block and pass control to your channel program. (When using the operating system, you cannot set the mode yourself.)

COMPLETION OF CHANNEL PROGRAM

The system considers the channel program completed when it receives an indication of a channel end condition. When channel end occurs, the request element for the channel program is made available, and a completion code is placed into the event control block. The completion code indicates whether errors are associated with channel end. If device end occurs simultaneously with channel end, errors associated with device end (i.e., unit exception or unit check) are also accounted for.

Device End Errors

If device end occurs after channel end and an error is associated with device end, the completion code in the event control block does not indicate the error. However, the status of the unit and channel is saved in the unit control block (UCB) for the device, and the UCB is marked as intercepted. The input/output block for the next request directed to the I/O device is also marked as intercepted. The error is assumed to be permanent, and the completion code in the event control block for the intercepted request indicates interception. The IFLGS field of the data control block is also flagged to indicate a permanent error. It should be noted that when a Write Tape Mark or Erase Long Gap CCW is the last (or only) CCW in your channel program, the I/O supervisor will not attempt recovery procedures for device end errors. In these circumstances, command chaining a NOPCCW to your Write Tape Mark or Erase Long Gap CCW ensures initiation of device end error recovery procedures.

To be prepared for device end errors, you should be familiar with device characteristics that can cause such errors. After one of your channel programs has terminated, you should not release buffer space

until you have determined that your next request for the device has not been intercepted. You may reissue an intercepted request.

INTERRUPTION HANDLING AND ERROR RECOVERY PROCEDURES

An I/O interruption allows the CPU to respond to signals from an I/O device which indicate either termination of a phase of I/O operations or external action on the device. A complete explanation of I/O interruptions is contained in the IBM System/360: Principles of Operation publication. For descriptions of interruptions by specific devices, refer to IBM Systems Reference Library publications for each device.

If error conditions are associated with an interruption, the input/output supervisor schedules the appropriate device-dependent error routine. The channel is then restarted with another request that is not related[1] to the channel program in error. If the error recovery procedures fail to correct the error, the system places ones in the first two bit positions of the IFLGS field of the data control block. You are informed of the error by an error code that the system places into the event control block.

Error Recovery Procedures for Related Channel Programs

Related channel programs are requests that are associated with a particular data control block and data extent block in the same job step. They must be executed in a definite order, i.e., the order in which the requests are received by the input/output supervisor. A channel program is not started until all previous requests for related channel programs have been completed. You specify, in the input/output block, whether the channel program is related to others.

If a permanent error occurs in a channel program that is related to other requests, the request elements for all the related channel programs are removed from their queue and made available. This process is called purging. The addresses of the input/output blocks for the related channel programs are chained together, with the address of the first input/output block in the chain placed into the "User Purge IOB Address" field of the data extent block. The address of the second input/output block is placed into the "Restart Address" field of the first input/output block, and so on. The last input/output block in the chain is indicated by all ones in its Restart Address field. The chain defines the order in which the request elements for the related channel programs are removed from the request queue.

For all requests that are related to the channel program in error, the system places completion codes into the event control blocks. The IFLGS field of the data control block is also flagged. Any requests for a data control block with error flags are posted complete without execution. If you wish to reissue requests that are related to the channel program in error, you must reset the first two bits of the IFLGS field of the data control block to zeros. You then issue a RESTORE macro instruction, specifying, as the only parameter, the address of the "User Purge IOB Address" field of the data extent block. This causes execution of all the related channel programs. (The RESTORE macro definition and how to add it to the macro-library are in the Appendix of this chapter.) Alternatively, if you wish to restart only particular channel programs rather than all of them, you may reissue the EXCP macro instruction for each channel program desired.

---

[1]Related channel programs are discussed in the next section.

This section discusses the appendages that you may optionally code when using the EXCP macro instruction. Before a programmer-written appendage can be executed, it must be included in the SVC library. These procedures are explained first; descriptions of the routines themselves and of their coding specifications follow.

DEFINING APPENDAGES

An appendage must be defined in a DD statement as a member of a SYS1 partitioned data set. The full member name of an appendage is eight bytes in length, but the first six bytes are required by IBM standards to be the characters IGG019. The last two characters must be provided by you as an identification; they may range in collating sequence from WA to Z9.

ENTERING APPENDAGES INTO SVC LIBRARY

The SVC library is a partitioned data set named SYS1.SVCLIB. You can insert an appendage into the SVC library during the system generation process or by link-editing it into the SYS1.SVCLIB. The routine must be a member of a cataloged partitioned data set whose name begins with SYS1.

To enter a routine into the SVC library during system generation, you use the SVCLIB macro instruction. The format of this macro instruction is given in the publication IBM System/360 Operating System: System Generation, GC28-6554.

CHARACTERISTICS OF APPENDAGES

An appendage is a programmer-written routine that provides additional control over I/O operations during channel program execution. By providing appendages, you can examine the status of I/O operations and determine the actions to be taken for various conditions. An appendage may receive control when one of the following occurs:

- Start I/O is issued.
- Program controlled interruption.
- End of extent.
- Channel end.
- Abnormal end.

Appendages are executed in supervisor state. You must not issue, in an appendage, any SVC instructions or instructions that change the status of the computing or operating system (e.g., WTO or LPSW). Since appendages are disabled for all types of interruptions except machine checks, you also must not enter loops that test for completion of I/O operations. An appendage must not alter storage used by either the supervisor or the input/output supervisor.

The identification of an appendage, which consists of the last two characters of its 8-character name, must be specified in the DCB macro instruction, as described in the section "EXCP Programming Specifications." When the OPEN macro instruction for the data control block is issued, any appendages specified in the DCB macro instruction are loaded into main storage. The appendages are linked to the input/output supervisor when their addresses are placed into a table of addresses called an appendage vector table. This table is always constructed by the system when OPEN is issued; if an appendage is not provided, the table contains the address of a return branch instruction to the input/output supervisor. Using the appendage vector table, the input/output supervisor branches and links to an appendage at the appropriate time. The address of the starting location of the appendage is placed into register 15.

Parameters are passed to appendages by the input/output supervisor. These parameters are contained in registers, and are as follows:

- Register 1: Address of the request queue element (RQE) for the channel program.

  The request queue element contains the following information:

  Bytes 1 and 2 -
  Link field when the RQE is an I/O queue.
  Bytes 3 and 4 -
  Address of the unit control block (UCB) for the I/O device.
  Byte 5 -
  Identification of the task control block (TCB) for the task. (In a multitasking environment, this field is not used. It contains all zeros if the request element is not available and all ones when the request element is available.)
  Bytes 6, 7, and 8 -
  Address of the input/output block.
  Byte 9 -
  Priority of the request, if the priority option has been selected for the system.
  Bytes 10, 11, and 12 -
  Address of the data extent block.

  The request queue element is normally 12 bytes in length; for a multitasking environment, it includes 4 more bytes that contain the address of the TCB.

- Register 2: Address of the input/output block (IOB).
- Register 3: Address of the data extent block (DEB).
- Register 4: Address of the data control block (DCB).
- Register 7: Address of the unit control block (UCB).
- Register 14: Address of the location in the input/output supervisor to which control is to be returned after execution of the appendage. When passing control from an appendage to the system, you may use displacements to the return address in register 14 for optional return procedures. Some of these procedures differ in their treatment of the request element associated with the channel program.
- Register 15: Address of the entry point to the appendage.

You may not change register 1 in an appendage; this is reserved in case an abnormal condition occurs while the appendage is in control. Register 9, if used, must be set to binary zero before control is returned to the system. All other registers, except those indicated in the descriptions of each appendage, must be saved and restored if they are used. The following table summarizes register conventions.

Appendages	Entry Point	Returns		Available Work Reg.
EOE	Reg 15	Reg 14 + 0 Reg 14 + 4 Reg 14 + 8	Extent Error Return Skip Try Again	Reg. 10, 11, 12 & 13
SIO	Reg 15	Reg 14 + 0 Reg 14 + 4	Normal Skip	Reg. 10, 11 & 13
PCI	Reg 15	Reg 14 + 0	Normal	Reg. 10, 11, 12 & 13
CE	Reg 15	Reg 14 + 0 Reg 14 + 4 Reg 14 + 8 Reg 14 + 12	Normal Skip Re-EXCP By-Pass	Reg. 10, 11, 12 & 13
XCE	Reg 15	Reg 14 + 0 Reg 14 + 4 Reg 14 + 8 Reg 14 + 12	Normal Skip Re-EXCP By-Pass	Reg. 10, 11, 12 & 13

The types of appendages are listed in the following paragraphs, with explanations of when they are entered, how they return control to the system, and which registers they may use without saving and restoring.

Start Input/Output (SIO) Appendage

This appendage is entered before the input/output supervisor issues a start input/output (SIO) instruction unless an error recovery procedure is in control. If SIO is not initiated because of a busy condition, the appendage will be reentered before SIO is reissued.

If the return address in register 14 is used to return control to the input/output supervisor, the I/O operation is executed normally. You may optionally bypass the SIO instruction and prevent execution of the channel program by using the contents of register 14 plus 4 as the return address. In this case, the channel program is not posted complete, but its request element is made available. You may do the posting by taking the following steps:

1. Save necessary registers.
2. Place pointer to post entry address from the CVT in Reg 15.
3. Place current TCB address from the RQE in Reg 12 for OS/MVT. For OS/MFT, place the TCB identifier from TCBIDF in Reg 12.
4. Place ECB address from the IOB in Reg 11.
5. Set the completion code in the high order byte in Reg 10.
6. Go to Post using BALR 14, 15.

You may use registers 10, 11, and 13 in a start input/output appendage without saving and restoring their contents.

Program Controlled Interruption (PCI) Appendage

This appendage is entered when a program controlled interruption occurs. At the time of the interruption, the contents of the channel status word will not have been placed in the "channel status word" field of the input/output block. The channel status word can be obtained from location 64. You must use the return address in register 14 to allow the system to proceed with normal interruption processing.

You may use registers 10 through 13 in a program controlled interruption appendage without saving and restoring their contents. This appendage may be reentered for the same channel program if the error recovery procedure is in the process of retrying a CCW with the program controlled bit set on. The IOBERR flag is set when the error recovery procedure is in control (IOBFL1 = X'20').

Refer to the topic "Block Multiplexer Channel Programming Notes" for special PCI conditions encountered with command retry.

End-of-Extent Appendage

This appendage is entered when the seek address specified in the input/output block is outside the allocated extent limits indicated in the data extent block.

If you use the return address in register 14 to return control to the system, the abnormal end appendage is entered. An end-of-extent error code (X'42') is placed in the "ECB code" field of the input/output block for subsequent posting in the event control block.

You may use the following optional return addresses:

- Contents of register 14 plus 4 - The channel program is posted complete; its request element is returned to the available queue.
- Contents of register 14 plus 8 - The request is tried again.

You may use registers 10 through 13 in an end-of-extent appendage without saving and restoring their contents.

Note: If an end-of-cylinder or file-protect condition occurs, the input/output supervisor updates the seek address to the next higher cylinder or track address, and re-executes the request. If the new seek address is within the data set's extent, the request is executed; if the new seek address is not within the data set's extent, the end-of-extent appendage is entered. If you wish to try the request in the next extent, you must move the new seek address into the UCB at UCB+48.

If a file protect condition occurs and was caused by a full seek (command code=07) embedded within a channel program, the request is flagged as a permanent error, and the abnormal end appendage is entered.

Channel End Appendage

This appendage is entered when a channel end, unit exception with or without channel end, or channel end with wrong length record occurs without any other abnormal end conditions.

If you use the return address in register 14 to return control to the system, the channel program is posted complete, and its request element is made available. In the case of unit exception or wrong length record, the error recovery procedure is performed before the channel program is posted complete, and the IOBEX flag (X'04') in IOBFL1 is set on. The condition code may be directly tested by using a BC instruction. A CC=0 means no UEX or WLR accompanied this interruption. The CSW status may be obtained from the IOBCSW.

If the appendage takes care of the wrong length record and/or unit exception it may turn off the IOBEX (X'04') flag in IOBFL1 and return normally. The event will then be posted complete (completion code 7F under normal conditions, taken from the high-order byte of the IOBECB field). If the appendage returns normally without resetting the IOBEX flag to zero, the request will be routed to the associated device error routine, and then the abnormal end appendage will be immediately entered. This abnormal end appendage will be entered with IOBECB completion code = '41'.

You may use the following optional return addresses:

- Contents of register 14 plus 4 - The channel program is not posted complete, but its request element is made available. You may post the event by using the calling sequence described under the start I/O appendage. This is especially useful if you wish to post an ECB other than the IOBECB.

- Contents of register 14 plus 8 - The channel program is not posted complete, and its request element is placed back on the request queue so that the I/O operation can be retried. For correct re-execution of the channel program, you must re-initialize the "Flags 1", "Flags 2", and "Flags 3" fields of the input/output block and set the "Error Counts" field to zero. As an added precaution, the IOBSNS and IOBCSW fields should be cleared.

- Contents of register 14 plus 12 - The channel program is not posted complete, and its request element is not made available. (The request element is assumed to be used in a subsequent asynchronous exit routine.)

You may use registers 10 through 13 in a channel end appendage without saving and restoring their contents.

Abnormal End Appendage

This appendage may be entered on abnormal conditions, such as: unit check, unit exception, wrong length indication, program check, protection check, channel data check, channel control check, interface control check, chaining check, out-of-extent error, and intercept condition (i.e., device end error). It may also be entered when an EXCP is issued for a DCB that has already been purged.

1. When this appendage is entered due to a unit exception and/or wrong length record indication, the IOBECB code is set to X'41'. For further information on these conditions see "Channel End Appendage."

2. When the appendage is entered due to an out-of-extent error, the IOBECB code is set to X'42'.

3. When this appendage is entered with the IOBECB code set to X'4B', it is because:

   a. The tape ERP having been entered after a repositioning for error recovery has been done and if there was a unit check, a check was made in the sense byte for load point, or

   b. The tape ERP determines that the IOBCSW command address contains zeros.

   An exit is made to the I/O supervisor with a permanent error indication.

4. When the appendage is first entered due to an intercept condition, the IOBECB code is set to X'7E'. If it is then determined that the error condition is permanent, the appendage will be entered a second time with the IOBECB code set to X'44'. The intercept condition signals that an error was detected at device end after channel end on the previous request.

5. When the appendage is entered due to an EXCP being issued to an already purged DCB, this request will enter the abnormal end appendage with the IOBECB code set to X'48'. This applies only to related requests.

6. When the appendage is entered with the IOBECB code set to 7F, it may be due to a unit check, program check, protection check, channel data check, channel control check, interface control check or chaining check. When the IOBECB code is 7F, it may be the first detection of an error in the associated channel program, or it could occur after an error routine has attempted to correct the error but was unsuccessful in its retry. Under these two conditions, the IOBERR flag is set; it indicates that the error routine is in control but has not yet declared the error to be permanent.

To determine if an error is permanent, you should check the "ECB code" field of the input/output block. To determine the type of error, check the channel status word and the sense information in the IOB. However, when the ECB code is X'42' or X'48', these fields are not applicable. For X'44' the CSW is applicable, but the sense is valid only if the unit check bit is set. If you use the return address in register 14 to return control to the system, the channel program is posted complete, and its request element is made available. (The SYNADAF macro instruction described in the Supervisor and Data Management Macro Instructions publication may be used in an error analysis routine to analyze permanent I/O errors.) You may use the following optional return addresses:

- Contents of register 14 plus 4 - The channel program is not posted complete, but its request element is made available.

- Contents of register 14 plus 8 - The channel program is not posted complete, and its request element is placed back on the request queue so that the request can be retried. For correct re-execution of the channel program, you must re-initialize the "Flags 1", "Flags 2", and "Flags 3" fields of the input/output block and set the "Error Counts" field to zero. As an added precaution, the IOBSNS and IOBCSW fields should be cleared.

- Contents of register 14 plus 12 - The channel program is not posted complete, and its request element is not made available. (The request element is assumed to be used in a subsequent asynchronous exit.)

You may use registers 10 through 13 in an abnormal end appendage without saving and restoring their contents.

Block Multiplexer Channel Programming Notes

Command retry is a new function of the block multiplexer channel supporting the 3330 Disk Storage and the 2305 Fixed Head Storage devices. When the channel receives a retry request, it repeats the execution of the channel command word (CCW) requiring no additional input/output interrupts. For example, a control unit may initiate a retry procedure to recover from a transient error.

A command retry during the execution of a channel program may cause any of the following conditions to be detected by the initiating program:

- Modifying CCWs: A CCW used in a channel program must not be modified before the CCW operation has been successfully completed. Without the command retry function, a command was fetched only once from storage by a channel. Therefore, a program could determine through condition codes or program controlled interruptions (PCI) that a CCW had been fetched and accepted by the channel. This permitted the CCW to be modified before re-execution. With the command retry function, this can not be done, since the channel will fetch the CCW from storage again on a command retry sequence. In the case of datachaining, the channel will command retry starting with the first CCW in the data chain.

- Program Controlled Interrupts: A CCW containing a PCI flag may cause multiple program controlled interruptions to occur. This happens if the PCI flagged CCW was retried during a command retry procedure, and a PCI could be generated each time the CCW is re-executed.

- Residual Count: If a channel program is prematurely terminated during the retry of a command, the residual count in the channel status word (CSW) will not necessarily indicate the extent of main storage used. For example, if the control unit detects a "wrong length record" error condition, an erroneous residual count is stored in the CSW until the command retry is successful. When the retry is successful, the residual in the CSW is the correct length of the data transfer. Since the channel will not allow more data to be transferred than is specified in the count field of the CCW, this situation will occur only when reading variable records or unknown record types.

- Command Address: When data chaining with command retry, the CSW may not indicate how many CCWs have been executed at the time of a PCI.

  For example:

CCW#	Channel Program	
1	Read	data chain
2	Read	data chain
3	Read	data chain, PCI
4	Read	command chain

  In this example, assume that the control unit signals command retry on Read #3 and the CPU accepts the PCI after the channel resets the command address to Read #1 because of command retry. The CSW stored for the PCI will contain the command address of Read #1, when actually the channel has progressed to Read #3.

- "Bit Spinning" on Data Read: Any program that tests a data storage location to determine when a CCW has been executed and continues to execute based on this data may get incorrect results if an error is detected and the CCW is retried. An example of this is a PCI appendage in which ones are placed in a buffer area that will be overlayed with zeros when a record is read. When the PCI appendage is entered, a check for zeros is made and the appendage will continue to loop until the record is read into the buffer (indicated by ones changed to zeros). If the appendage uses the data from this record to modify a channel program, the results will be unpredictable during a command retry sequence, as the CCW has not been correctly executed.

EXCP Programming Specifications

This section describes the parameters of the macro instructions that you must use with EXCP, and the fields of the required control blocks.

MACRO INSTRUCTIONS

If you are using the EXCP macro instruction you must also use DCB, OPEN, CLOSE, and, in some cases, the EOV macro instruction. The parameters of these macro instructions, and of the EXCP macro instruction itself, are listed and explained here. A diagram of the data control block is included with the description of the DCB macro instruction.

DCB -- Define Data Control Block for EXCP

The EXCP form of the DCB macro instruction produces a data control block that can be used with the EXCP macro instruction. You must issue a DCB macro instruction for each data set to be processed by your channel programs. Notation conventions and format illustrations of the DCB macro instruction are given in the Data Management Macro Instructions publication. DCB parameters that apply to EXCP may be divided into four categories, depending on the following portions of the data control block that are generated when they are specified:

- Foundation block. This portion is required and is always 12 bytes in length. You must specify two of the parameters in this category.
- EXCP interface. This portion is optional. If you specify any parameter in this category, 20 bytes are generated.
- Foundation block extension and common interface. This portion is optional and is always 20 bytes in length. If this portion is generated, the device dependent portion is also generated.
- Device dependent. This portion is optional and is generated only if the foundation block extension and common interface portion is generated. Its size ranges from 4 to 20 bytes, depending on specifications in the DEVD parameter of this category. However, if you do not specify the DEVD parameter (and the foundation extension and common interface portion is generated), the maximum 20 bytes for this portion are generated.

Some of the procedures performed by the system when the data control block is opened and closed (such as writing file marks for output data sets on direct access volumes) require information from optional data control block fields. You should make sure that the data control block is large enough to provide all information necessary for the procedures you want the system to handle.

Figure EXCP1 shows the relative position of each portion of an opened data control block. The fields corresponding to each parameter of the DCB macro instruction are also designated, with the exception of DDNAME, which is not included in a data control block that has been opened. The fields identified in parentheses represent system information that is not associated with parameters of the DCB macro instruction.

Sources of information for data control block fields other than the DCB macro instruction are data definition (DD) statements, data set labels, and data control block modification routines. You may use any of these sources to specify DCB parameters. However, if a portion of the data control block is not generated by the DCB macro instruction, the system does not accept information intended for that portion from any alternative source.

FOUNDATION BLOCK PARAMETERS:

DDNAME=symbol
The name of the data definition (DD) statement that describes the data set to be processed. This parameter must be given.

MACRF=(E)
The EXCP macro instruction is to be used in processing the data set. This parameter must be given. When creating, updating, or accessing a direct (BDAM) data set that resides on more than one volume, you should code the DCB macro instruction using MACRF=(R) or MACRF=(W) instead of MACRF=(E) so that the operating system will automatically mount all of the volumes of the data set and complete the DEB.

REPOS= Y
N

Magnetic tape volumes:
If your system generation statements include the dynamic device reconfiguration (DDR) entry, then this parameter controls whether the DDR routine will attempt to reposition the volume after swapping devices. (To have the DDR routine attempt to reposition your tape volume, you must maintain the block count in the DCBBLKCT field.)

Y - Yes, attempt to reposition.

N - No, do not attempt to reposition.

If the entry is omitted, N is assumed.

EXCP INTERFACE PARAMETERS:

EOEA=symbol
2-byte identification of an end-of-extent appendage that you have entered into the SVC library. (See Note A.)

PCIA=symbol
2-byte identification of a program controlled interruption (PCI) appendage that you have entered into the SVC library. (See Note A.)

SIOA=symbol
2-byte identification of a start I/O (SIO) appendage that you have entered into the SVC library. (See Note A.)

CENDA=symbol
2-byte identification of a channel end appendage that you have entered into the SVC library. (See Note A.)

XENDA=symbol
2-byte identification of an abnormal end appendage that you have entered into the SVC library. (See Note A.)

OPTCD=code
A code of Z indicates that for magnetic tape (input only) a reduced error recovery procedure (5 reads only) will occur when a data check is encountered. It should be specified only when the tape is known to contain errors and the application does not require that all records be processed. Its proper use would include error frequency analysis in the SYNAD routine. Specification of this parameter will also cause generation of a foundation block extension. This parameter is ignored unless it was selected at system generation.

IMSK=value
Any specification indicates that the system will not use IBM supplied error routines.

Note A: The full name of an appendage is eight bytes in length, but the first six bytes are required by IBM standards to be the characters IGG019. You provide the last two characters as the 2-byte identification; they may range in collating sequence from WA to Z9.

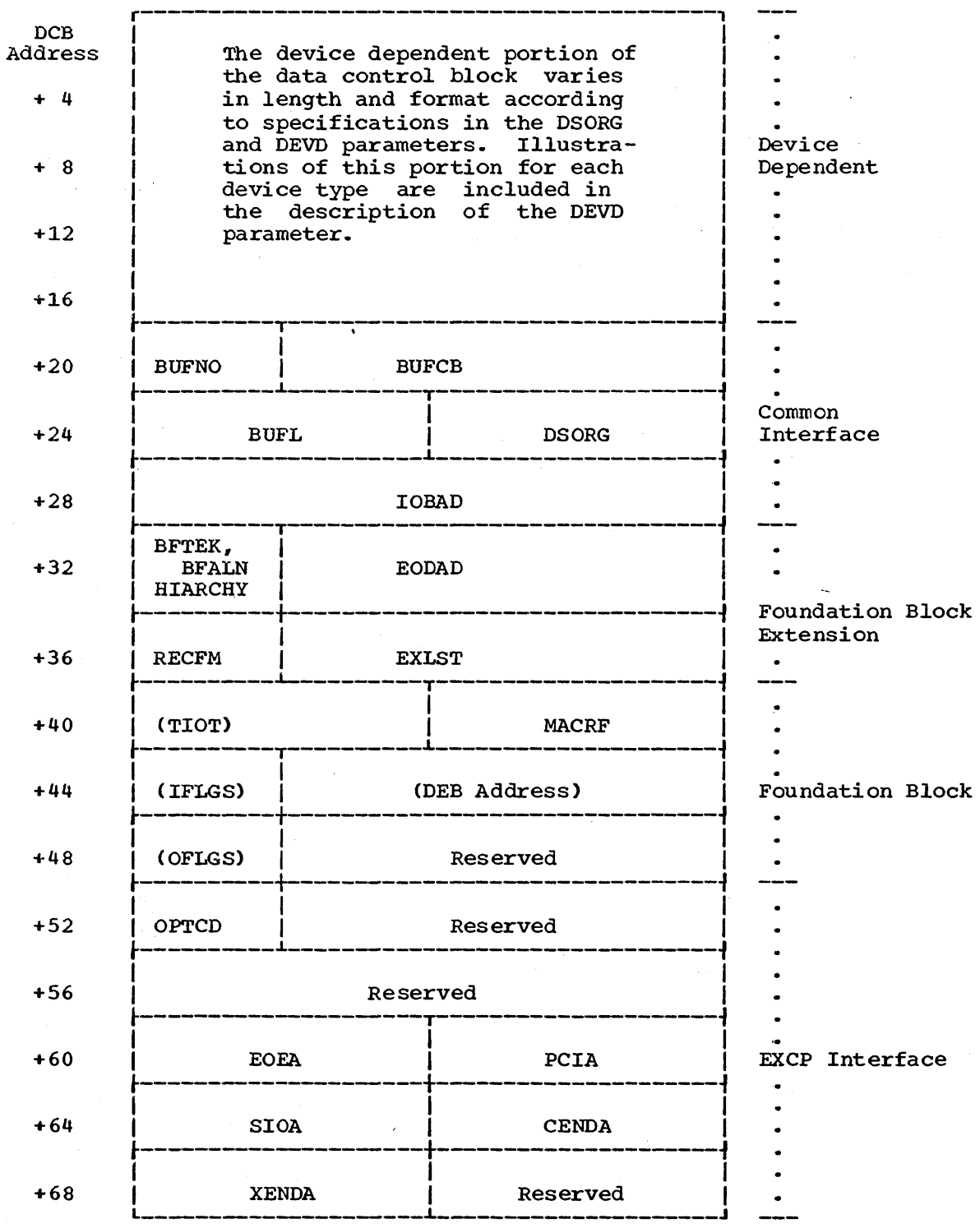


Figure EXCP1. Data Control Block Format for EXCP (After OPEN)

FOUNDATION BLOCK EXTENSION AND COMMON INTERFACE PARAMETERS:

EXLST=relexp
specifies the address of an exit list that you have written for exceptional conditions. The format of this exit list is given in the Data Management Services publication.

EODAD=relexp
specifies the address of your end-of-data set routine. If this routine is not available when it is required, the task is abnormally terminated.

DSORG=code
specifies the data set organization as one of the following codes. Each code indicates that the format of the device dependent portion of the data control block is to be similar to that generated for a particular access method:

Code	DCB Format for
PS	QSAM or BSAM
PO	BPAM
DA	BDAM
IS	QISAM or BISAM

Note: For direct access devices, if you specify either PS or PO, you must maintain the following fields of the device dependent portion of the data control block so that the system can write a file mark for output data sets:

- The track balance (TRBAL) field, which contains a 2-byte binary number that indicates the remaining number of bytes on the current track.
- The full disk address (FDAD-MBBCCHHR) field, which indicates the location of the current record.

IOBAD=relexp
specifies the address of an input/output block (IOB). If a pointer to the current IOB is not required, you may use this field for any purpose.

The following parameters are not used by the EXCP routines but provide cataloging information about the data set. This information can be used later by access method routines that read or update the data set.

RECFM=code
specifies the record format of the data set. Record format codes are given in the Data Management Macro Instructions publication. When writing a data set to be read later without EXCP, the RECFM, LRECL, and BLKSIZE should be specified to identify the data set attributes. LRECL and BLKSIZE can only be specified in a JCL statement, since these fields do not exist in a DCB used by EXCP.

BFTEK={S|E}
specifies the buffer technique as either simple or exchange. BFTEK bits 0 and 5 specify whether hierarchy 0 or hierarchy 1 is used to form the buffer pool. If HIARCHY={0|1} is omitted from the DCB, the buffer pool is formed in hierarchy 0.

BFALN={F|D}
specifies the word boundary alignment of each buffer as either fullword or doubleword.

BUFL=absexp
specifies the length in bytes of each buffer; the maximum length is 32,767.

BUFNO=absexp
specifies the number of buffers assigned to the associated data set; the maximum number is 255.

BUFCB=relexp
specifies the address of a buffer pool control block, i.e., the 8-byte field preceding the buffers in a buffer pool.

DEVICE DEPENDENT PARAMETERS:

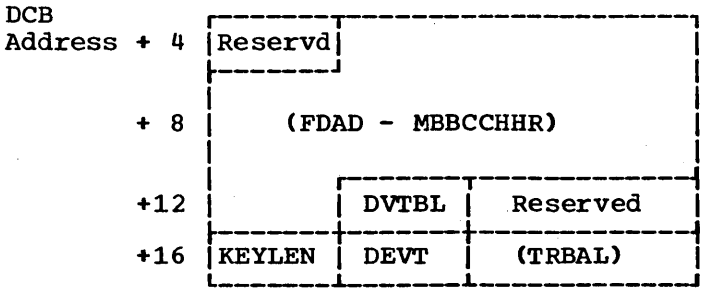
DEVD=code
specifies the device on which the data set may reside as one of the following codes. The codes are listed in order of descending space requirements for the data control block:

Code	Device
DA	Direct access
TA	Magnetic tape
PT	Paper tape
PR	Printer
PC	Card punch
RD	Card reader

Note: If you do not wish to select a specific device until job set up time, you should specify the device type requiring the largest area.

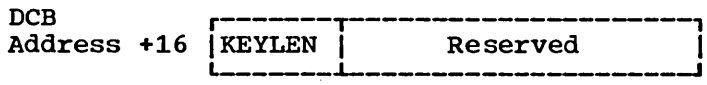
The following diagrams illustrate the device dependent portion of the data control block for each device type specified in the DEVD parameter, and for each data set organization specified in the DSORG parameter. Fields that correspond to device dependent parameters in addition to DEVD are indicated by the parameter name. For special services, you may have to maintain the fields shown in parentheses. The special services are explained in the note that follows the diagram.

Device dependent portion of data control block when DEVD=DA and DSORG=PS or PO:

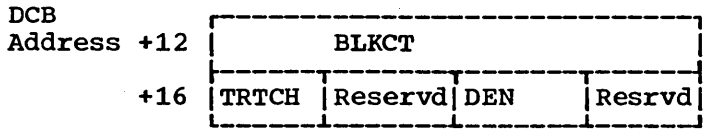


Note: For output data sets, the system uses the contents of the full disk address (FDAD-MBBCCHHR) field plus one to write a file mark when the data control block is closed, provided the track balance (TRBAL) field indicates that space is available. You must maintain the contents of these two fields yourself if the system is to write a file mark. OPEN will initialize DVTBL and DEVT.

Device dependent portion of data control block when DEVD=DA and DSORG=IS or DA:



Device dependent portion of data control block when DEVD=TA and DSORG=PS:

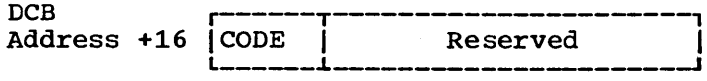


Note: For output data sets, the system uses the contents of the block count (BLKCT) field to write the block count in trailer labels when the data control block is closed, or when the EOV macro instruction is issued. You must maintain the contents of this field yourself if the system is to write the correct block count.

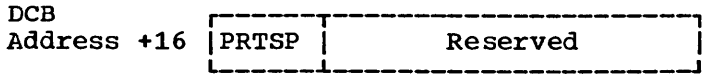
When using EXCP to process a tape data set open at a checkpoint, you must be careful to maintain the correct count; otherwise the system may position the data set incorrectly when restart occurs.

If your system generation statements include the dynamic device reconfiguration entry, this field must be maintained by you for repositioning. Also, your DCB macro instruction must include the REPOS=Y entry.

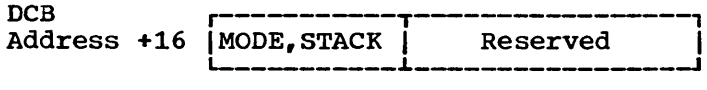
Device dependent portion of data control block when DEVD=PT and DSORG=PS:



Device dependent portion of data control block when DEVD=PR and DSORG=PS:



Device dependent portion of data control block when DEVD=PC or RD and DSORG=PS:



The following parameters pertain to specific devices and may be specified only when the DEVD parameter is specified.

KEYLEN=value
specifies, for direct access devices, the length in bytes of the key of a physical record, with a maximum value of 255. When a block is read or written, the number of bytes transmitted is the key length plus the record length.

CODE=value
specifies, for paper tape, the code in which records are punched as follows:

Value	Code
I	IBM BCD
F	Friden
B	Burroughs
C	National Cash Register
A	ASCII
T	Teletype
N	no conversion (format F records only)

If this parameter is omitted, N is assumed.

DEN=value
specifies, for magnetic tape, the tape recording density in bits per inch as follows:

Value	Density	
	Model 2400/3400 7-track	Model 2400/3400 9-track
0	200 (2400 only)	-
1	556	-
2	800	800
3	-	1600

If this parameter is omitted, the lowest density is assumed.

TRTCH=value
specifies, for 7-track magnetic tape, the tape recording technique as follows:

Value	Tape Recording Technique
C	Data conversion feature is available.
E	Even parity is used. (If omitted, odd parity is assumed.)
T	BCDIC to EBCDIC translation is required.

MODE=value
specifies, for a card reader or punch, the mode of operation. Either C (column binary mode) or E (EBCDIC code) may be specified.

STACK=value
specifies, for a card punch or card reader, the stacker bin to receive cards as either 1 or 2.

PRTSP=value
specifies, for a printer, the line spacing as either 0, 1, 2, or 3.

OPEN -- Initialize Data Control Block

The OPEN macro instruction initializes one or more data control blocks so that their associated data sets can be processed. You must issue OPEN for all data control blocks that are to be used by your channel programs. (A dummy data set may not be opened for EXCP.) Some of the procedures performed when OPEN is executed are:

- Construction of data extent block (DEB).
- Transfer of information from DD statements and data set labels to data control block.
- Verification or creation of standard labels.
- Tape positioning.
- Loading of programmer-written appendage routines.

The three parameters of the OPEN macro instruction are:

dcb-addr
specifies the address of the data control block to be initialized. (More than one data control block may be specified.)

opt_1
specifies the intended method of I/O processing of the data set. You may specify this parameter as either INPUT, RDBACK, or OUTPUT. For each of these, label processing when OPEN is executed is as follows:

INPUT - Header labels are verified.
RDBACK - Trailer labels are verified.
OUTPUT - Header labels are created.

If this parameter is omitted, INPUT is assumed.

opt_2
specifies the volume disposition that is to be provided when volume switching occurs. The operand values and meanings are as follows:

REREAD	Reposition the volume to process the data set again.
LEAVE	No additional positioning is performed at end-of-volume processing.
DISP	The disposition indicated on the DD statement is tested and appropriate positioning provided. This service is assumed if this operand is omitted and volume positioning is applicable. If there is no disposition specified in the DD statement when this operand is specified, LEAVE is assumed.

OPEN processes the EXCP macro as a combination of the DSORG specified in the DCB and physical sequential (PS). The DSORG specified in the DCB only affects the merging of certain DSCB, JFCB, and DCB fields. For all other operations, EXCP is considered to have a DSORG of PS. Only one volume is processed for EXCP by the OPEN routine, except when there are concatenated partitioned data sets, in which case all volumes concerned are processed. Thus, direct access and indexed sequential organized data sets will have only the first volume processed by OPEN.

The list and execute forms of the OPEN macro instruction are described in the Data Management Macro Instruction publication.

EXCP -- Execute Channel Program

The EXCP macro instruction requests the initiation of the I/O operations of a channel program. You must issue EXCP whenever you want to execute one of your channel programs. The only parameter of the EXCP macro instruction is:

iob-addrx
specifies the address, or a register that contains the address of the input/output block of the channel program to be executed.

EOV -- End of Volume

The EOV macro instruction identifies end-of-volume and end-of-data set conditions. For an end-of-volume condition, EOV causes switching of volumes and verification or creation of standard labels. For an end-of-data set condition, EOV causes your end-of-data set routine to be entered. Before processing trailer labels on a tape input data set, you must decrement the DCBBLKCT field. You issue EOV if switching of magnetic tape or direct access volumes is necessary, or if secondary allocation is to be performed for a direct access data set opened for output.

For magnetic tape, you must issue EOV when either a tapemark is read or a reflective spot is written over. In these cases, bit settings in the 1-byte OFLGS field of the data control block determine the action to be taken when EOV is executed. Before issuing EOV for magnetic tape, you must make sure that appropriate bits are set in OFLGS. Bit positions 2,3,6, and 7 of OFLGS are used only by the system; you are concerned with bit positions 0,1,4, and 5. The use of these OFLGS bit positions is as follows:

Bit 0
indicates that a tape mark is to be written.
Bit 1
indicates that a backwards read was the last I/O operation.
Bit 4
indicates that data sets of unlike attributes are to be concatenated.
Bit 5
indicates that a tape mark has been read.

If bits 0 and 5 of OFLGS are both off when EOV is executed, the tape is spaced past a tapemark, and standard labels, if present, are verified on both the old and new volumes. The direction of spacing depends on bit 1. If bit 1 is off, the tape is spaced forward; if bit 1 is on, the tape is backspaced.

If bit 0 is on when EOV is executed, a tapemark is written immediately following the last data record of the data set, standard labels, if specified, are created on the old and the new volume.

When issuing EOV for sequentially organized output data sets on direct access volumes, you can determine whether additional space has been obtained on the same or a different volume. You do this by checking the volume serial number in the unit control block (UCB) both before and after issuing EOV.

The only parameter of the EOV macro instruction is:

dcb-addrx
specifies the address of the data control block that is opened for the data set. If this parameter is specified as (1), register 1 must contain this address.

CLOSE -- Restore Data Control Block

The CLOSE macro instruction restores one or more data control blocks so that processing of their associated data sets can be terminated. You must issue CLOSE for all data control blocks that were used by your channel programs. Some of the procedures performed when CLOSE is executed are:

- Release of data extent block (DEB).
- Removal of information transferred to data control block fields when OPEN was executed.
- Verification or creation of standard labels.
- Volume disposition.
- Release of programmer-written appendage routines.

The two parameters of the CLOSE macro instruction are:

dcb-addr
specifies the address of the data control block to be restored. More than one data control block may be specified.

opt
specifies the type of volume disposition intended for the data set. You may specify this parameter as either LEAVE or REREAD. The corresponding volume disposition when CLOSE is executed is as follows:

LEAVE - Volume is positioned at logical end of data set.
REREAD - Volume is positioned at logical beginning of data set.
DISP - The disposition indicated on the DD statement is tested, and appropriate positioning is provided. This service is assumed if this operand is omitted and volume positioning is applicable. If there is no disposition specified in the DD statement when this operand is specified, LEAVE is assumed.

This parameter is ignored if specified for volumes other than magnetic tape or direct access.

Note: When CLOSE is issued for data sets on magnetic tape volumes, labels are processed according to bit settings in the OFLGS field of the data control block. Before issuing CLOSE for magnetic tape, you must set the appropriate bits in OFLGS. The OFLGS bit positions that you are concerned with are listed in the EOV macro instruction description.

The list and execute forms of the CLOSE macro instruction are described in the Data Management Macro Instructions publication.

CONTROL BLOCK FIELDS

The fields of the input/output block, event control block, and data extent block are illustrated and explained here; the data control block fields have been described with the parameters of the DCB macro instruction in the section "EXCP Programming Specifications."

Input/Output Block Fields

The input/output block is not automatically constructed by a macro instruction; it must be defined as a series of constants and must be on a fullword boundary. For unit record and tape devices, the input/output block is 32 bytes in length. For direct access, teleprocessing, and graphic devices, 8 additional bytes must be provided.

In Figure EXCP2, the shaded areas indicate fields in which you must specify information. The other fields are used by the system and must be defined as all zeros. You may not place information in these fields, but you may examine them.

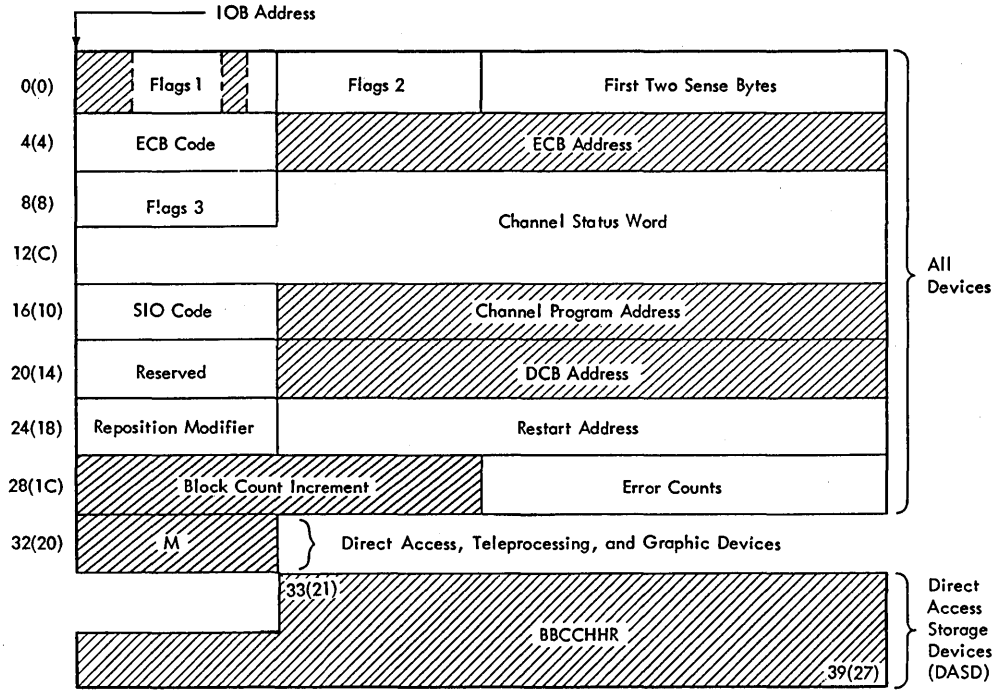


Figure EXCP2. Input/Output Block Format

Flags 1 (1 byte)
specifies the type of channel program. You must set bit positions 0, 1, and 6. One bits in positions 0 and 1 indicate data chaining and command chaining, respectively. (If both data chaining and command chaining are specified, the system does not use error recovery routines except for the 2311, 2671, 1052, and 2150.) A one bit in position 6 indicates that the channel program is not related to any other channel program. Bit positions 2, 3, 4, 5, and 7 are used only by the system.

Flags 2 (1 byte)
is used only by the system.

First two sense bytes (2 bytes)
are placed into the input/output block by the system when a unit check occurs.

ECB code (1 byte)
indicates the first byte of the completion code for the channel program. The system places this code in the high order byte of the event control block when the channel program is posted complete. The completion codes and their meanings are listed under "Event Control Block Fields."

ECB address (3 bytes)
specifies the address of the 4-byte event control block that you have provided.

Flags 3 (1 byte)
is used only by the system.

Channel status word (7 bytes)
indicates the low order seven bytes of the channel status word, which are placed into this field each time a channel end occurs.

SIO code (1 byte)
indicates in bits 0 and 1 the instruction-length code, in bits two and three the condition code for the SIO instruction that the system issues to start the channel program, and in bits 4 through 7 the program mask.

Channel program address (3 bytes)
specifies the starting address of the channel program to be executed.

Reserved (1 byte)
is used only by the system.

DCB address (3 bytes)
specifies the address of the data control block of the data set to be read or written by the channel program.

Reposition modifier (1 byte)
is used by the system for volume repositioning in error recovery procedures.

Restart address (3 bytes)
is used by the system to indicate the starting address of a channel program that performs special functions for error recovery procedures. The system also uses this field in procedures for making request elements available, as explained under "Error Recovery Procedures for Related Channel Programs."

Block count increment (2 bytes)
specifies, for magnetic tape, the amount by which the block count (BLKCT) field in the device dependent portion of the data control block is to be incremented. You may alter these bytes at any time. For forward operations, these bytes should contain a binary positive integer (usually + 1); for backward operations, they should contain a binary negative integer. When these bytes are not used, all zeros must be specified.

Error counts (2 bytes)
indicates the number of retries attempted during error recovery procedures.

M (1 byte)
Direct access devices:
Extent entry in the data extent block that is associated with the channel program. (0 indicates the first extent; 1 indicates the second, etc.)
Teleprocessing and graphic devices:
The UCB index.

BBCCHHR (7 bytes)
specifies, for direct access devices, the seek address for the programmer's channel program.

Event Control Block Fields

You must define an event control block as a 4-byte area on a fullword boundary. When the channel program has been completed, the input/output supervisor places a completion code containing status information into the event control block (Figure EXCP3). Before examining this information, you must test for the setting of the "Complete Bit." If the complete bit is not on, and the problem program cannot perform other

useful operations, you should issue a WAIT macro instruction that specifies the event control block. Under no circumstances may you construct a program loop that tests for the complete bit.

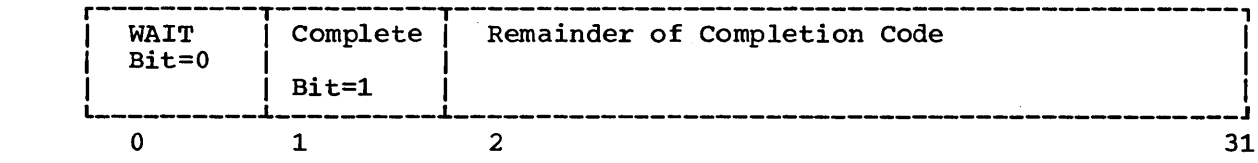


Figure EXCP3. Event Control Block After Posting of Completion Code

WAIT bit
A one bit in this position indicates that the WAIT macro instruction has been issued, but that the channel program has not been completed.

Complete bit
A one bit in this position indicates that the channel program has been completed; if it has not been completed, a zero bit is in this position.

Completion code
This code, which includes the WAIT and complete bits, may be one of the following 4-byte hexadecimal expressions:

Code	Interpretation
7F000000	Channel program has terminated without error.
41000000	Channel program has terminated with permanent error.
42000000	Channel program has terminated because a direct access extent address has been violated.
44000000	Channel program has been intercepted because of permanent error associated with device end for previous request. You may reissue the intercepted request.
48000000	Request element for channel program has been made available after it has been purged.
4B000000	One of the following errors occurred during tape error recovery processing: • The CSW command address in the IOB is zeros. • An unexpected load point was encountered.
4F000000	Error recovery routines have been entered because of direct access error but are unable to read home address or record 0.
50000000	Channel program terminated with error. Input block was a DOS embedded checkpoint record.

Data Extent Block Fields

The data extent block is constructed by the system when an OPEN macro instruction is issued for the data control block. You may not modify the fields of the data extent block, but you may examine them. The data extent block format and field description is contained in the System Control Block publication.

Appendix: RESTORE and PURGE Macro Instructions

If you want to use the RESTORE or PURGE macro instruction, you must either add the macro definitions to the macro-library (SYS1.MACLIB) or place them in a separate partitioned data set and concatenate this data set to the macro-library. This section contains the following:

- The format of the macro instruction.
- The job control and utility statements needed to add the macro definition to the library.
- The macro definition to be added to the library.

RESTORE Macro Instruction

This macro instruction is used to return purged request elements to the request queues. The format of this macro instruction is as follows:

Name	Operation	Operand
	RESTORE	User purge IOB address

The user purge IOB address is the address of a pointer to the first IOB address in a previously purged IOB list. It could be the DEBUSRPG field in the data extent block (see "SVC Purge Routine").

RESTORE Macro Definition

```
            MACRO
&NAME       RESTORE         &LIST
            AIF             ('&LIST' EQ '').E1
&NAME       IHBINNRA        &LIST                   LOAD REG 1
            SVC             17                      ISSUE SVC FOR RESTORE
            MEXIT
.E1         IHBERMAC        01,150                  LIST ADDR MISSING
            MEND
```

Control Statements Required

```
//jobname       JOB     {parameters}
//stepname      EXEC    PGM=IEBUPDTE,PARM=NEW
//SYSPRINT      DD      SYSOUT=A
//SYSUT2        DD      DSNAME=SYS1.MACLIB,DISP=OLD
//SYSIN         DD      DATA
./        ADD   NAME=RESTORE,LIST=ALL
                .
                .
                .
                RESTORE Macro definition
                .
                .
                .
./        ENDUP
/*
```

PURGE Macro Instruction

The PURGE macro instruction is used to return request elements to the I/O supervisor inactive queue (next available).

PURGE Macro Definition

```
            MACRO
&NAME       PURGE           &LIST
            AIF             ('&LIST'EQ'').E1
&NAME       IHBINNRA        &LIST                   LOAD REG 1
            SVC             16
            MEXIT
.E1         IHBERMAC        01,147                  LIST ADDR MISSING
            MEND
```

Control Statements Required

```
//jobname       JOB     {parameter}
//stepname      EXEC    PGM=IEBUPDTE,PARM=NEW
//SYSPRINT      DD      SYSOUT=A
//SYSUT2        DD      DSNAME=SYS1.MACLIB,DISP=OLD
//SYSIN         DD      *
./        ADD   NAME=PURGE,LIST=ALL
                .
                .
                PURGE Macro definition
                .
                .
./        ENDUP
/*
```

Name	Operation	Operand
symbol	PURGE	User purge parameter list

The purge parameter list is constructed in the user's program area. Depending on the options specified in the PURGE parameter list, elements can be purged from

1. The asynchronous exit queue of the task supervisor.
2. The request blocks chained to the TCB.
3. The I/O supervisor logical channel queues.

You can bypass the purge of the RBs chained to the TCB by setting bit 5 of the option byte. The parameter list is constructed prior to issuing the PURGE macro instruction; this list must fall on a fullword boundary. It is either a three-word list or, if bit 4 of the options byte in Word 1 equals one (1), a four-word list. It is constructed as follows:

Word 1

Byte 1
(options byte)

Bit 0 - Specified DEB or DEB chain =0 - Purge request elements associated with complete DEB chain starting at the DEB specified in bytes 2, 3, and 4 of word 1. =1 - Purge only the request elements associated with the DEB specified by bytes 2, 3, and 4 of word 1.
Bit 1 - POST request purged or ignore posting. =0 - Do not POST the purged requests. =1 - POST the purge requests, code = X'48'.
Bit 2 - HALT I/O or quiesce active requests. =0 - Allow the active requests to quiesce. =1 - HALT the I/O operations. (The HALT I/O is simulated if the operation is a SEEK.
Bit 3 - Purge all or only related requests. =0 - Purge all requests. =1 - Purge only related requests.
Bit 4 - Normal purge or list purge. =0 - Normal purge. =1 - Purge TCB list.
Bit 5 - Purge all queues or bypass RB purge. =0 - Purge AEQ, RB, and I/O supervisor logical channel queues. =1 - Purge only the I/O supervisor logical channel queue(s) and AEQ.
Bit 6 - Purge by TCB or DEB =0 - Purge by DEB =1 - Purge by TCB Note: This bit must be zero in order to honor bit 0. If this bit is one, all requests associated with the TCB are purged, and bit 0 is ignored.
Bit 7 - (Spare)

Bytes 2, 3, and 4
DEB address - not required if purging by TCB.

Word 2

Byte 1
completion code

Bytes 2, 3, and 4
TCB address - if none, the current TCB is used.

Word 3

Byte 1

Quiesce indicator field. It will indicate X'01' if one or more requests are quiescing.

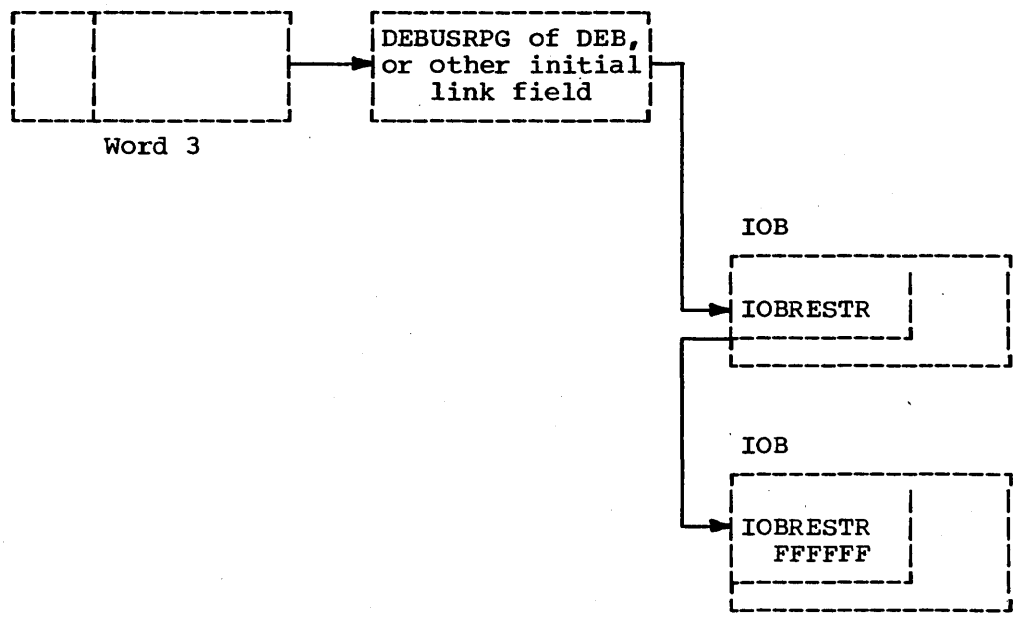
Bytes 2, 3, and 4

Address of the initial link field for chaining IOBs that are purged. The initial link field can be the user purge field in the DEB (DEBUSRPG) or any area you select. The initial link field points to the first IOB in the chain. At the completion of purge, the contents of word 3 are unpredictable. No chaining is done when TCB with HALT I/O option is specified.

If the IOB restart field (IOBRESTR) is used as a link field, the last one will contain X'FFFFFF' in its three low-order bytes.

The following figure shows the IOB chain.

Chaining IOBs



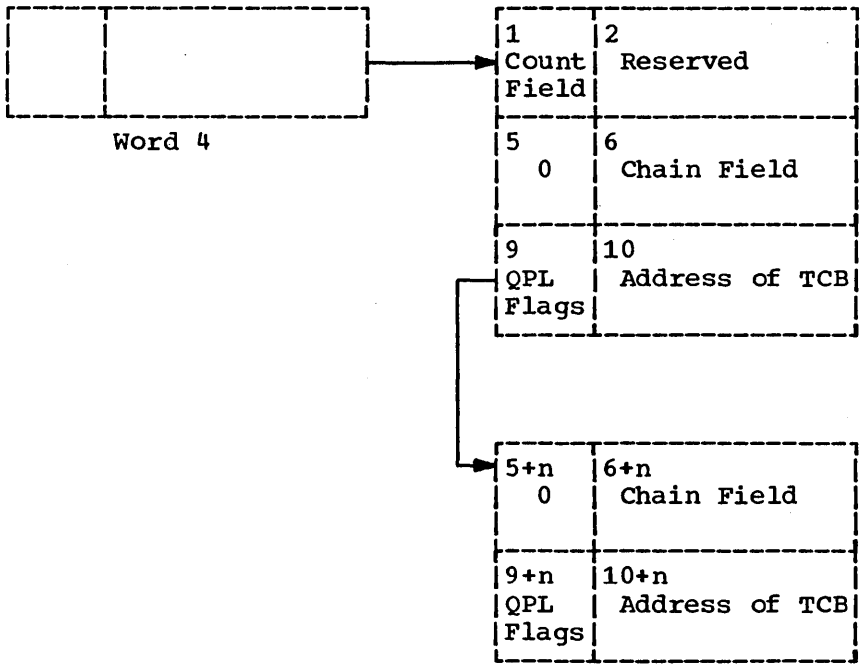
IOB Chain for PURGE

Word 4

Byte 1
(flag byte)

Bit 1 - Purge or wait flag. =0 - Purge entry. =1 - Wait entry.
Bit 2 - Wait flag. =0 - Return to caller before waiting. =1 - Perform purge and wait operations, and do not return to caller.
Bits 3-8 - Reserved.

Bytes 2,3, and 4
Address of the QUIESCE I/O parameter list (QPL). This field points to a list of TCBs that are to be purged. The format of the list is shown below.



n = 8x(# of TCBs to be purged -1)

1 Count field.

A temporary count field used to keep track of the number of TCBs that have been purged.

6 Chain field.

Address of the initial link field for chaining IOBs that are purged. See the illustration for chaining IOBs in this section.

7 QPL flags - Last entry or current entry.

Bit 0 - Last entry flag.
=0 - More entries follow.
=1 - Last entry.

Bit 1 - Current entry flag.
=0 - Not current.
=1 - Current.

Bits 2-8 - Reserved.

8 TCB address.

Address of the TCB to be purged.

ATLAS--Assign an Alternate Track and Copy Data From the Defective Track

A program that uses the EXCP macro instruction for input and output may use the ATLAS macro instruction, during the execution of the program, to obtain an alternate track and to copy a defective track onto the alternate track. With the use of ATLAS, the program can recover from permanent (hard) errors encountered in the execution of the following types of I/O commands:

- Search ID.
- Write.
  (The error condition must be confirmed during the execution of the channel program by a CCW that checks the data written.)
- Read count.
  Errors in the CCHHR part of the count area can be recovered from unless the record is the home address or record zero.

  Errors in the KDD part of the count area cannot be recovered from unless the user has identified the defective record.

Your DCB must include the DCBRECFM field and the field must show whether the data set is in the track overflow format. If it is, recovery from errors in last records on tracks depends on your identifying the track overflow record segments.

Recovery takes the form of obtaining an alternate good track and copying the defective track onto the good alternate one. Unless a re-execution of the channel program by ATLAS can correct the defect, the user should examine, and if necessary replace, defective records in a subsequent job if the data set is to be processed again.

ATLAS MACRO INSTRUCTION

The format of the macro instruction is:

Name	Operation	Operands
(symbol)	ATLAS	PARMADR={address / (register)} [,CHANPRG={R / NR}] [,CNTPTR={P / F}] [,WRITS={YES / NO}]

PARMADR
Address of a parameter address list of the following format:

Address of the parameter address list

+0	Address of the IOB for the channel program that encountered the error
+4	Main storage address of the count area field

The count area field contains the CCHHRKDD of a defective record or the CCHH of a track that is to be copied.

address - Address is given as the symbolic label of the address list.

(register) - Address is given as the number of a general register (1-12) that contains the address of the list.

CHANPRG
Condition of the channel program that encountered the error.

R - Channel program may be re-executed by ATLAS. Before permitting re-execution of the channel program by ATLAS, you must reset the error indications of the previous execution fields in the DCBIFLGS. (See the example of the use of ATLAS below.)
NR - Channel program may not be re-executed.

If this parameter is omitted, R is assumed.

,CNTPTR
Contents of the count area field.

P - Part of the count area - the CCHH address of the track to be copied.
F - Full count area - CCHHRKDD count of the record found defective.

If this parameter is omitted, P is assumed.

,WRITS
track overflow segment identification.

If your data set is in the track overflow format, this identification determines recovery from errors in last records on tracks.

YES - If this is the last record on the track, it is a segment other than the last of a track overflow record.
NO - If this is the last record on the track, it is the last or only segment of a track overflow record.

If this parameter is omitted, it is assumed that it cannot be established whether a last record is a segment of an overflow record.

USE OF ATLAS

If a channel program encounters a unit check condition (shown in the CSW) in its execution, the I/O supervisor program will place the Sense bytes in the IOB. ATLAS can be used to recover from Sense conditions shown by the following bit settings:

IOBSENS0	X'08'	Data check (except in the count area)
IOBSENS1	X'80'	Data check in the count area
IOBSENS1	X'02'	Missing address marker (But see the following for combinations of this bit setting for which ATLAS is powerless.)

However, defects in the home address record or the record zero record cannot be recovered from through the use of ATLAS. These conditions are shown by:

IOBSENS1 X'02' and IOBSENS0 X'01' - home address defect.

IOBSENS1 X'0A' - record zero defect, or, home address cannot be located.

Also, before using ATLAS, you must reset error indications as follows:

```
NI    DCBIFLGS,X'3F'    Reset the DCBIFLGS error indications.
```

The ATLAS program will attempt to find a good alternate track and will attempt to copy the defective track onto the good track, including all error conditions in either key or data areas. The error conditions may be rectified by re-executing the channel program or through the use of the IEHATLAS utility program in a subsequent step.

The following illustrates the use of the ATLAS macro instruction.

```
        EXCP    MYIOB
        WAIT    ECB=MYECB
        TM      MYECB,X'20'          TEST FOR I/O ERROR
        BO      NEXT                 NO, SUCCESSFUL, GO TO ANOTHER
                                     ROUTINE
        TM      IOBCSW+3,X'02'       UNIT CHECK
        BL      OTHER                NO, DO OTHER ERROR PROCESSING
        TM      IOBSENS0,X'08'       DATA CHECK
        BO      ATLASGO              YES, VALID ERROR
        TM      IOBSENS1,X'80'       DATA CHECK IN COUNT
        BO      ATLASGO              YES, VALID ERROR
        TM      IOBSENS1,X'0A'       MISSING ADDRESS MARKER
        BO      OTHER                YES, ATLAS CANNOT HANDLE ERROR
                                     DO OTHER ERROR PROCESSING
ATLASGO EQU     *
        NI      DCB1FLGS,X'3F'       RESET ERROR INDICATORS
        ATLAS   PARMADR=THERE,CHANPRG=R
```

OPERATION OF THE ATLAS PROGRAM

The ATLAS program (SVC 86):

- Establishes the availability and address of the next alternate track from the format 4 DSCB of the VTOC.
- Brings all count fields from the defective track into main storage to establish the description of the track.
- Initializes the alternate track. (Write home address, write record zero.)
- Brings the key and data areas of each record into main storage, one at a time, and combines them with their new count area to write the complete record onto the alternate track.
- When the copying is finished, chains the alternate to the defective track and updates the VTOC.

RETURN CODES

When control returns to the user, he will find one of the following decimal return codes in register 15: (Note that for return codes 0, 36, 40, and 44 the contents of register 0 may be significant.)

Decimal Return Code	Meaning
0	Successful completion. Key and data areas have been copied from the defective track onto a good alternate one. The only error encountered was in the record identified by the user's CCHHRKDD value. If the channel program is re-executable, it has been successfully re-executed.
4	This device type (2301 drum, 2303 drum) does not have alternate tracks that can be assigned by programming.
8	All alternate tracks for the device have been assigned.
12	A request for storage (GETMAIN macro instruction) could not be satisfied.
16	All attempts to initialize and transfer data to an alternate track failed. The number of attempts made is equal to 10% of the assigned alternates for the device.
20	The type of error shown by the sense byte cannot be handled through the use of the ATLAS macro instruction. The condition is other than a data check (in the count or data areas) or a missing address marker.
24	The format 4 DSCB of the VTOC cannot be read, therefore alternate track information is not available to ATLAS.
28	The record specified by the user was the format 4 DSCB and it could not be read.
32	An error found in count area of last record on the track cannot be handled because last-record-on-track identification is not supplied.
36	An error was encountered reading or writing the home address record or reocrd zero. No error recovery has taken place. If register 0 contains X'01 00 00 00', the defect is in record zero.
40	Successful completion. Key and data areas have been copied from the defective track onto a good alternate one. However, the alternate track may have records with defective key or data areas. Register 0 identifies the first three found defective as follows:  n - Number of record numbers that follow (0, 1, 2, or 3). R - The number of the record found defective but copied anyhow. If the channel program is re-executable, it has been successfully re-executed.

44 - Error/Errors encountered and no alternate track has been assigned. The return parameter register (RO) will contain the R of a maximum of three error records.

Error conditions that return this code are:

1. ATLAS received an error indication for a record with a data length in the count field of zero. Recovery was not possible because a distinction cannot be made between an EOF record and an invalid data length.

2. An error occurred while reading the count field of a record and the KDD (key length-data length) was found to be defective.

3. More than three records on the specified track contained errors in their count fields.

48 - No errors found on the track, no alternate assigned. ATLAS will not assign an alternate unless a track has at least one defective record.

52 - I/O error in re-executing user's channel program. A good alternate is chained to the defective track and data has been transferred. The user's control blocks will give indication of the error condition causing failure in re-execution of his channel program.

56 - The DCB reflects a track overflow data set but the UCB device type shows that the device does not support track overflow.

60 - The CCHH of the user specified count area is not within the extents of his data set.

Figures EXCP4 and EXCP5 summarize the return codes that reflect track error conditions by error location.

Record in Error		Area in Error			
		Count Area		Key Area	Data Area
		CCHHR	KDD		
Record r ($r \neq 0$)					
Not Last on Track		0	44	40	40
Last on Track	WRITS=YES	0	44	40	40
	WRITS=NO	0	44	40	40
	Omitted*	32	44	40	40
Record Zero					
		36	36	36	36
Home Address					
		36			

* Omitted and the Data Set is in the Track Overflow Format.

Figure EXCP4. Error Locations and Return Codes if CCHH is in the Count Area Field

Record in Error		Area in Error			
		Count Area		Key Area	Data Area
		CCHHR	KDD		
Record n (n=R in CCHHRKDD)					
Not Last on Track		0	0	0	0
Last on Track	WRITS=YES	0	0	0	0
	WRITS=NO	0	0	0	0
	Omitted *	32	32	0	0
Record m (m ≠ R in CCHHRKDD)					
Not Last on Track		0	44	40	40
Last on Track	WRITS=YES	0	44	40	40
	WRITS=NO	0	44	40	40
	Omitted *	32	44	40	40
Record Zero					
		36	36	36	36
Home Address					
		36			

* Omitted and the Data Set is in the Track Overflow Format.

Figure EXCP5. Error Locations and Return Codes if CCHHRKDD is in the Count Area Field

Execute Direct Access Program (XDAP) Macro Instruction

This chapter explains what the execute direct access program (XDAP) macro instruction does and how you can use it. The control block generated when XDAP is issued and the macro instructions used with XDAP are also discussed.

The XDAP macro instruction provides you with a means of reading, verifying, or updating blocks on direct access volumes without using an access method and without writing your own channel program. Since most of the specifications for XDAP are similar to those for the execute channel program (EXCP) macro instruction, it is recommended that you be familiar with the "EXCP Macro Instruction" chapter of this publication, as well as with the information contained in the required publication.

PREREQUISITE PUBLICATION

The IBM System/360 Operating System: Supervisor Services and Macro Instructions publication (GC28-6646) explains the standard procedures for I/O processing under the operating system.

Execute Direct Access Program (XDAP) Macro Instruction

Execute direct access program (XDAP) is a macro instruction of System/360 Operating System that you may use to read, verify, or update a block on a direct access volume. If you are not using the standard IBM data access methods, you can, by issuing XDAP, generate the control information and channel program necessary for reading or updating the records of a data set.

You cannot use XDAP to add blocks to a data set, but you can use it to change the keys of existing blocks. Any block configuration and any data set organization can be read or updated.

Although the use of XDAP requires much less main storage space than do the standard access methods, it does not provide many of the control program services that are included in the access methods. For example, when XDAP is issued, the system does not block or deblock records and does not verify block length.

To issue XDAP, you must provide the actual device address of the track containing the block to be processed. You must also provide either the block identification or the key of the block, and specify which of these is to be used to locate the block. If a block is located by identification, both the key and data portions of the block may be read or updated. If a block is located by key, only the data portion can be processed.

Requirements for Execution of Direct Access Program

Before issuing the XDAP macro instruction, you must issue a DCB macro instruction, which produces a data control block (DCB) for the data set to be read or updated. You must also issue an OPEN macro instruction, which initializes the data control block and produces a data extent block (DEB).

When the XDAP macro instruction is issued, another control block, containing both control information and executable code, is generated. This control block may be logically divided into three sections:

- An event control block (ECB), which is supplied with a completion code each time the direct access channel program is terminated.
- An input/output block (IOB), which contains information about the direct access channel program.
- A direct access channel program, which consists of three channel command words (CCWs). The type of channel program generated depends on specifications in the parameters of the XDAP macro instruction.

After this XDAP control block is constructed, the direct access channel program is executed. A block is located by either its actual address or its key, and is either read or updated.

When the channel program has terminated, a completion code is placed into the event control block. After issuing XDAP, you should therefore issue a WAIT macro instruction specifying the event control block to determine whether the direct access program has terminated. If volume switching is necessary, you must issue an EOV macro instruction. When processing of the data set has been completed, you must issue a CLOSE macro instruction to restore the data control block.

MACRO INSTRUCTIONS

When you are using the XDAP macro instruction, you must also issue DCB, OPEN, CLOSE, and, in some cases, the EOV macro instruction. The parameters of the XDAP macro instruction are listed and described here. For the other required macro instructions, special requirements or options are explained, but you should refer to the "EXCP Macro Instruction" section of this publication for listings of their parameters.

DCB -- Define Data Control Block

The EXCP form of the DCB macro instruction produces a data control block that can be used with the XDAP macro instruction. You must issue a DCB macro instruction for each data set to be read or updated by the direct access channel program. The "EXCP Macro Instruction" section of this publication contains a diagram of the data control block, as well as a listing of the parameters of the DCB macro instruction.

OPEN -- Initialize Data Control Block

The OPEN macro instruction initializes one or more data control blocks so that their associated data sets can be processed. You must issue OPEN for all data control blocks that are to be used by the direct access program. Some of the procedures performed when OPEN is executed are:

- Construction of data extent block (DEB).
- Transfer of information from DD statements and data set labels to data control block.
- Verification or creation of standard labels.
- Loading of programmer-written appendage routines.

The two parameters of the OPEN macro instruction are the address(es) of the data control block(s) to be initialized, and the intended method of I/O processing of the data set. The method of processing may be specified as either INPUT or OUTPUT; however, if neither is specified, INPUT is assumed.

XDAP -- Execute Direct Access Program

The XDAP macro instruction produces the XDAP control block (i.e., the ECB, IOB, and channel program) and executes the direct access channel program. The format of the XDAP macro instruction is:

Operation	Operand
XDAP	ecb-symbol,type-{R|W|V}{I|K},dcb-addr,area-addr ,length-value,[(key-addr,keylength-value)],blkref-addr [,sector-addr]

ecb-symbol
specifies the symbolic name to be assigned to the XDAP control block.

type-{R|W|V}{I|K}
specifies the type of I/O operation intended for the data set and the method by which blocks of the data set are to be located.

The codes and their meanings are as follows:

R - Read a block.
W - Write a block.
V - Verify contents of a block but do not transfer data.
I - Locate a block by identification. (The key portion, if present, and the data portion of the block are read or written.)
K - Locate a block by key. (Only the data portion of the block is read or written.)

dcb-addr
specifies the address of the data control block of the data set.

area-addr
specifies the address of an input or output area for a block of the data set.

length-value
specifies the number of bytes to be transferred to or from the input or output area. If blocks are to be located by identification and the data set contains keys, the value must include the length of the key. The maximum number of bytes transferred is 32767.

key-addr
specifies, when blocks are to be located by key, the address of a main storage field that contains the key of a block to be read or overwritten.

keylength-value
specifies, when blocks are to be located by key, the length of the key. The maximum length is 255 bytes.

blkref-addr
specifies the address of a main storage field containing the actual device address of the track containing the block to be located. When blocks are to be located by key, this field is seven bytes in length; when blocks are to be located by identification, an eighth byte indicating block identification must be included in this field. (The actual address of a block is in the form MBBCCHHR, where M indicates which extent entry in the data extent block is associated with the direct access program; BB indicates the bin number of direct access volume; CC indicates the cylinder address; HH indicates the actual track address; and R indicates the block identification.)

sector-addr
specifies the address of a one-byte field containing a sector value. The sector-address parameter is used for rotational position sensing (RPS) devices only. The parameter is optional, but its use will improve channel performance. When the parameter is coded, a set-sector CCW (using the sector value indicated by the data address field) precedes the Search-ID-Equal command in the channel program. The sector-address parameter is ignored if the type parameter is coded as RK, WK, or VK. If a sector address is specified in the list form of the macro, then a sector address (not necessarily the same) must be specified in the execute form.
Note: No validity check is made on either the address or the sector value when the XDAP macro is issued. However, a unit exception interrupt will occur during the channel program execution if the sector value is larger than the maximum for the device or if the macro is issued against a device without RPS.

EOV -- End of Volume

The EOV macro instruction identifies end-of-volume and end-of-data set conditions. For an end-of-volume condition, EOV causes switching of volumes and verification or creation of standard labels. For an end-of-data set condition, EOV causes your end-of-data set routine to be entered. When using XDAP, you issue EOV if switching of direct access volumes is necessary, or if secondary allocation is to be performed for a direct access data set opened for output.

The only parameter of the EOV macro instruction is the address of the data control block of the data set.

CLOSE -- Restore Data Control Block

The CLOSE macro instruction restores one or more data control blocks so that processing of their associated data sets can be terminated. You must issue CLOSE for all data sets that were used by the direct access channel program. Some of the procedures performed when CLOSE is executed are:

- Release of data extent block (DEB).
- Removal of information transferred to data control block fields when OPEN was executed.
- Verification or creation of standard labels.
- Release of programmer-written appendage routines.

The only parameter of the CLOSE macro instruction is the address of the data control block to be restored. (More than one data control block may be specified.)

THE XDAP CONTROL BLOCK

The three portions of the control block generated during execution of the XDAP macro instruction are described here.

Event Control Block (ECB)

The event control block begins on a full word boundary and occupies the first 4 bytes of the XDAP control block. Each time the direct access channel program terminates, the input/output supervisor places a completion code containing status information into the event control block (Figure XDAP1). Before examining this information, you must test for the setting of the "Complete Bit" by issuing a WAIT macro instruction specifying the event control block.

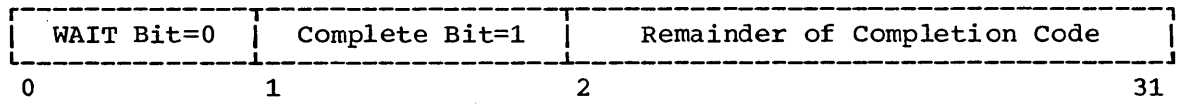


Figure XDAP1. Event Control Block After Posting of Completion Code

WAIT Bit
: A one bit in this position indicates that the WAIT macro instruction has been issued, but that the direct access channel program has not been completed.

Complete Bit
: A one bit in this position indicates that the channel program has been completed; if it has not been completed, a zero bit is in this position.

Completion Code

This code, which includes the WAIT and complete bits, may be one of the following 4-byte hexadecimal expressions:

Code	Interpretation
7F000000	Direct access program has terminated without error.
41000000	Direct access program has terminated with permanent error.
42000000	Direct access program has terminated because a direct access extent address has been violated.
44000000	Channel program has been intercepted because of permanent error associated with device end for previous request. You may reissue the intercepted request.
48000000	Request element for channel program has been made available after it has been purged.
4B000000	One of the following errors occurred during tape error recovery processing: • The CSW command address in the IOB is zeros. • An unexpected load point was encountered.
4F000000	Error recovery routines have been entered because of direct access error but are unable to read home address or record 0.
50000000	Channel program terminated with error. Input block was a DOS embedded checkpoint record.

Input/Output Block (IOB)

The input/output block is 40 bytes in length and immediately follows the event control block. The section "EXCP Macro Instruction" of this publication contains a diagram of the input/output block. The only fields with which the user of XDAP is concerned are the "First Two Sense Bytes" and "Channel Status Word" fields. You may wish to examine these fields when a unit check condition or an I/O interruption occurs.

Direct Access Channel Program

The direct access channel program is 24 bytes in length and immediately follows the input/output block. Depending on the type of I/O operation specified in the XDAP macro instruction, one of four channel programs may be generated. The three channel command words for each of the four possible channel programs are shown in Figure XDAP2.

When a sector address is specified with an RI, VI, or WI operation, the channel program is 32 bytes long. In Figure XDAP2 each of the channel programs would be preceded by a Set Sector command.

Type of I/O Operation	CCW	Command Code
Read by identification	1	Search ID Equal
	2	Transfer in Channel
Verify by identification[1]	3	Read Key and Data
Read by key	1	Search Key Equal
	2	Transfer in Channel
Verify by key[1]	3	Read Data
Write by identification	1	Search ID Equal
	2	Transfer in Channel
	3	Write Key and Data
Write by key	1	Search Key Equal
	2	Transfer in Channel
	3	Write Data
[1]For verifying operations, the third CCW is flagged to suppress the transfer of information to main storage.		

Figure XDAP2. The XDAP Channel Programs

XDAP Options

CONVERSION OF RELATIVE TRACK ADDRESS TO ACTUAL ADDRESS

To issue XDAP, you must provide the actual device address of the track containing the block to be processed. If you know only the relative track address, you can convert it to the actual address by using a resident system routine. The entry point to this conversion routine is labeled IECPCNVT. The address of the entry point is in the communication vector table (CVT). The address of the CVT is in location 16. (The CVT macro instruction defines the symbolic names of all fields in the CVT. The macro definition and how to add it to the macro-library are in the Appendix of this chapter.)

The conversion routine does all its work in general registers. You must load registers 0, 1, 2, 14, and 15 with input to the routine. Register usage is as follows:

Register	Use
0	Must be loaded with a 4-byte value of the form TTRN, where TT is the number of the track relative to the beginning of the data set, R is the identification of the block on that track, and N is the concatenation number of the data set. (0 indicates the first or only data set in the concatenation, 1 indicates the second, etc.)
1	Must be loaded with the address of the data extent block (DEB) of the data set.
2	Must be loaded with the address of an 8-byte area that is to receive the actual address of the block to be processed. The converted address is of the form MBBCCHHR, where M indicates which extent entry in the data extent block is associated with the direct access

program (0 indicates the first extent, 1 indicates the second, etc.); BB indicates the bin number of the direct access volume; CC indicates the cylinder address; HH indicates the actual track address; and R indicates the block identification.

3-8	Are not used by the conversion routine.
9-13	Are used by the conversion routine and are not restored.
14	Must be loaded with the address to which control is to be returned after execution of the conversion routine.
15	Is used by the conversion routine as a base register and must be loaded with the address at which the conversion routine is to receive control.

OBTAINING SECTOR NUMBER OF A BLOCK ON AN RPS DEVICE

To obtain the performance improvement given by rotational position sensing, you should specify the sector-addr parameter on the XDAP macro. For programs which may be used for both RPS and non-RPS devices, the UCBTYP field can be checked to determine whether or not the device has the rotational position sensing feature.

The sector-addr parameter on the XDAP macro specifies the address of a one byte field in your region. You must store the sector number of the block to be located in this field. You can obtain the sector number of the block by using a resident conversion routine, IEC0SCR1. The address of this routine is in field CVT0SCR1 of the CVT, and the address of the CVT is in location 16. The routine should be invoked via a BALR 14, 15 instruction.

The conversion routine does all its work in general registers. You must load registers 0, 1, 2, 14, and 15 with input to the routine. Register usage is as follows:

Register	Use
0	For fixed length records, register 0 must be loaded with a 4-byte value of the form DDKR, where DD is a 2-byte field containing the physical block size, K is a 1-byte field containing the key length, and R is a 1-byte field containing the record number for which a sector value is desired. The high-order bit of register 0 must be turned off to indicate fixed-length records. For variable length records, register 0 must be loaded with a 4-byte value in the form of BBIR, where BB is a 2-byte field containing the total number of key and data bytes up to but not including the target record, I is a 1-byte key indicator (1 for keyed records, 0 for non-keyed records), and R is a 1-byte field containing the record number for which a sector value is desired. The high order bit of register 0 must be turned on to indicate variable length records.
1	Not used by the sector convert routine.
2	Must be loaded with a 4-byte field in which the first byte is the UCB device type code for the device (obtainable from UCB+19), and the remaining three bytes are the address of a 1-byte area that is to receive the sector value.
3-8,12,13	Not used.
9-11	Used by the convert routine and are not saved or restored.
14	Must be loaded with the address to which control is to be returned after execution of the sector conversion routine.
15	Used by the conversion routine as a base register and must be loaded with the address of the entry point to the conversion routine.

APPENDAGES

For additional control over I/O operations, you may write appendages, which must be entered into the SVC library. Descriptions of these routines and their coding specifications are contained in the "EXCP Macro Instruction" section of this publication.

L- AND E-FORMS OF XDAP MACRO INSTRUCTION

You may use the L-form of the XDAP macro instruction for a macro expansion consisting of only a parameter list, or the E-form for a macro expansion consisting of only executable instructions.

Note: The BLKREF parameter is ignored by the "L" form of the XDAP macro instruction. The field may be supplied in the E-form of the macro instruction or moved into the IOB by you.

Appendix: CVT Macro Instruction

If you want to use the CVT macro instruction, you must add the macro definition to the macro-library (SYS1.MACLIB). This section contains the following:

- The format of the CVT macro instruction.
- The job control and utility statements needed to add the macro definition to the library.

The fields in the communication vector table are described in the System Control Blocks publication.

Use the DEVTYPE macro instruction to request information relating to the characteristics of an I/O device, and to cause this information to be placed in a specified area. (The results of a DEVTYPE macro instruction executed before a checkpoint is taken should not be considered valid after a checkpoint/restart occurs.)

Format of the CVT Macro Instruction

This macro instruction defines the symbolic names of all fields in the communication vector table (CVT). When coding this macro instruction, you must precede it with a DSECT statement. The format of the macro instruction is as follows:

Name	Operation	Operand
	CVT	

Control Statements Required

```
//jobname       JOB     {parameters}
//stepname      EXEC    PGM=IEBUPDTE,PARM=NEW
//SYSPRINT      DD      SYSOUT=A
//SYSUT2        DD      DSNAME=SYS1.MACLIB,DISP=OLD
//SYSIN         DD      *
./        ADD   NAME=CVT,LIST=ALL
                .
                .
                .
                CVT Macro definition
                .
                .
                .
./        ENDUP
/*
```

Data Set Protection

To use the data set protection feature of the operating system, you must create and maintain a password data set consisting of records that associate the names of the protected data sets with the password assigned to each data set.

There are three ways to maintain the PASSWORD data set:

- You can write your own routines.
- You can use the PROTECT macro instruction.
- You can use the utility control statements of the IEHPROGM utility program.

This chapter is divided into two sections. The first section describes the general features of data set protection, including the use of your own routines to maintain the password data set. It provides the information you need to create the data set and characteristics of the data set. The second section discusses the PROTECT macro; it provides the programming information you need to use the macro and discusses the difference between using the PROTECT macro and using your own routines to maintain the password data set.

PSWD

RECOMMENDED PUBLICATIONS

The IBM System/360 Operating System: Data Management Services publication (GC28-3746) contains a general description of the data set protection feature.

The IBM System/360 Operating System: Messages and Codes publication (GC28-6631) contains a description of the operator messages and replies associated with the data set protection feature.

The IBM System/360 Operating System: Job Control Language Reference publication (GC28-6704) contains a description of the data definition (DD) statement parameter used to indicate that a data set is to be placed under protection.

The IBM System/360 Operating System: Direct Access Device Space Management program logic manual, (GY28-6607) contains a description of the password data set record format.

The IBM System/360 Operating System: Utilities publication (GC28-6586) contains a description of how to maintain the PASSWORD data set using the utility statements of the IEHPROGM utility program.

Implementing Data Set Protection

To prepare for use of the data set protection feature of the operating system, you place a sequential data set, named PASSWORD, on the system residence volume (containing SYS1.NUCLEUS and SYS1.SVCLIB). Note: If the routines that you write to maintain the password data set use the basic direct access method (BDAM), you must place a BDAM data set named PASSWORD on the system residence volume. This data set must contain one record for each data set placed under protection. In turn, each record contains a data set name, the password for that data set, a counter field, a protection mode indicator, and a field for recording any information you desire to log. On the system residence volume, these records are formatted as a "key area" (data set name and password) and a "data area" (counter field, protection mode indicator, and logging field). The data set is searched on the "key area."

You can write routines to create and maintain the PASSWORD data set. (If you use the PROTECT macro instruction to maintain the password data set, see the section in this chapter called USING THE PROTECT MACRO INSTRUCTION TO MAINTAIN THE PASSWORD DATA SET. If you use the IEHPROGM utility program to maintain the PASSWORD data set, see the publication IBM System/360 Operating System: Utilities, GC28-6586.) These routines may be placed in your own library or the system's linkage editor library (SYS1.LINKLIB). You may use a data management access method or EXCP programming to handle the PASSWORD data set.

If a data set is to be placed under protection, it must have a protection indicator set in its label (DSCB or header 1 tape label). This is done by the operating system when the data set is created or by the IEHPROGM utility program. The protection indicator is set in response to an entry in the LABEL= parameter of the DD statement associated with the data set being placed under protection. The Job Control Language Reference publication describes the entry. Note: Data sets on magnetic tape are protected only when standard labels are used.

Users who wish to have the password supplied by some method other than operator key-in may replace the password reading module with their own routine. The READPSWD source module may be used as a base for writing a new module. In this case, the new object module replaces module READPSWD on the SVCLIB.

The balance of this chapter discusses the PASSWORD data set characteristics, the creation of protected data sets, and operating characteristics of the data set protection feature.

Password Data Set Characteristics

The PASSWORD data set must reside on the same volume as your operating system. The space you allocate to the PASSWORD data set must be contiguous, i.e., its DSCB must indicate only one extent. The amount of space you allocate is dependent on the number of data sets your installation desires to place under protection. Each entry in the PASSWORD data set requires 132 bytes of space. The organization of the PASSWORD data set is physical sequential, the record format is unblocked, fixed length records (RECFM=F). These records are 80 bytes long (BLKSIZE=80) and form the data area of the PASSWORD data set records on direct access storage. In these direct access storage records, the data area is preceded by a key area of 52 bytes (KEYLEN=52). The password assigned may be from one to eight alphameric characters. The IBM System/360 Operating System Direct Access Device Space Management program logic manual (GY28-6607) describes the password data set record format.

Protecting the Password Data Set

You can protect the PASSWORD data set itself by creating a password record for it when your program initially builds the data set. Thereafter, the PASSWORD data set cannot be opened (except by the operating system routines that scan the data set) unless the operator enters the password.

Note: If a problem occurs on a password-protected system data set, maintenance personnel must be provided with the password in order to access the data set and resolve the problem.

Creating Protected Data Sets

A data definition (DD) statement parameter (LABEL=) is used to indicate that a data set is to be placed under protection. You may create a data set, and set the protection indicator in its label, without entering a password record for it in the PASSWORD data set. However, once the data set is closed, any subsequent opening results in termination of the program attempting to open the data set, unless the password record is available and the operator can honor the request for the password. Operating procedures at your installation must ensure that password records for all data sets currently under protection are entered in the PASSWORD data set.

Protection Feature Operating Characteristics

This section provides information concerning actions of the protection feature in relation to termination of processing, volume switching, data set concatenation, SCRATCH and RENAME functions, and counter maintenance.

Termination of Processing

Processing is terminated when:

1. The operator cannot supply the correct password for the protected data set being opened after two tries.
2. A password record does not exist in the PASSWORD data set for the protected data set being opened.
3. The protection mode indicator setting in the password record, and the method of I/O processing specified in the open routine do not agree, e.g., OUTPUT specified against a read-only protection mode indicator setting.
4. There is a mismatch in data set names for a data set involved in a volume switching operation. This is discussed in the next section.

Volume Switching

The operating system end-of-volume routine does not request a password for a data set involved in a volume switch. Continuity of protection is handled in the following ways:

Input Data Sets - Tape and Direct Access Devices
Processing continues if there is an equal comparison between the data set name in the tape label or DSCB on the volume switched to, and the name of the data set opened with the password. An unequal comparison terminates processing.

Output Data Sets - Tape Devices
The protection indicator in the tape label for the first data set on the volume switched to is tested:

1. If the protection indicator is set ON, an equal comparison between the data set name in the label and the name of the data set opened with the password allows processing to continue. An unequal comparison results in a call for another volume.
2. If the protection indicator is OFF, processing continues, and a new label is written with the protection indicator set ON.
3. If only a volume label exists on the volume switched to, processing continues, and a new label is written with the protection indicator set on.

Output Data Sets - Direct Access Devices
For existing data sets, an equal comparison between the data set name in a DSCB on the volume switched to, and the name of the data set opened with the password allows processing to continue. For new output data sets, the mechanism used to effect volume switching ensures continuity of protection and the DSCB created on the new volume will indicate protection.

Data Set Concatenation

A password is requested for every protected data set that is involved in a concatenation of data sets, regardless of whether the other data sets involved are protected or not.

SCRATCH and RENAME Functions

An attempt to perform the SCRATCH or RENAME functions on a protected data set results in a request for the password. The protection feature issues an operator's message (IEC301A) when a protected data set is the object of these functions. The password supplied when attempting to scratch or rename a data set must be a WRITE password. The Messages and Codes publication discusses the message.

Counter Maintenance

The operating system does not maintain the counter in the password record and no overflow indication will be given (overflow after 65,535 openings). You must provide a counter maintenance routine to check and, if necessary, reset this counter.

Using the Protect Macro Instruction to Maintain the Password Data Set

To use the PROTECT macro instruction, your password data set must be on the system residence volume. The PROTECT macro can be used to:

- Add an entry to the password data set.
- Replace an entry in the password data set.
- Delete an entry from the password data set.
- Provide a list of information about an entry in the password data set; this list will contain the security counter, access type, and the 77 bytes of security information in the "data area" of the entry.

In addition, the PROTECT macro, will update the DSCB of the protected data set, for a direct access device, to reflect its protected status; this feature eliminates the need for you to use job control language whenever you place a data set under protection.

PASSWORD DATA SET CHARACTERISTICS AND RECORD FORMAT WHEN YOU USE THE PROTECT MACRO

When you use the PROTECT macro, the record format and characteristics of the password data set should be the same as the record format and characteristics when you use your own routines to maintain it, with two exceptions: the number of records that you establish for each protected data set and the values of the protection mode indicator.

Number of Records for Each Protected Data Set: When you use the PROTECT macro, the password data set must contain at least one record for each protected data set. The password (the last 8 bytes of the "key area") that you assign when you place the data set under protection for the first time is called the control password, in addition, you may create as many secondary records for the same protected data set as you need. The passwords assigned to these additional records are called secondary passwords. This feature is helpful if you want several users to have access to the same protected data set, but you also want to control the manner in which they can use it. For example: one user could be assigned a password that allowed the data set to be read and written, and another user could be assigned a password that allowed the data set to be read only.

Note: The PROTECT macro will update the DSCB of the protected data set only when you issue it for adding, replacing or deleting a control password.

Protection Mode Indicator: You can set the protection mode indicator in the password record to four different values:

- X'00' to indicate that the password is a secondary password and the protected data set is to be read only.
- X'80' to indicate that the password is the control password and the protected data set is to be read only.
- X'01' to indicate that the password is a secondary password and the protected data set is to be read and written.
- X'81' to indicate the password is the control password and the protected data set is to be read and written.

Since the DSCB of the protected data set is updated only when the control password is changed, it is possible to request protection attributes for secondary passwords which conflict with the protection attributes of the control password.

If the control password has read only protection, its secondary passwords may have read only or read write protection. A request for a secondary password with read without password protection will result in a secondary password with read write protection. A read only control password may be changed to a read write control password without affecting any secondary passwords, but if a read only control password is changed to a read without password control password all secondary passwords will automatically become read without password secondary passwords.

If the control password has read write protection, its secondary passwords may have read only or read write protection. A request for a secondary password with read without password protection will result in a secondary password with read write protection. A read.write control password may be changed to a read only control password without affecting any secondary passwords, but if a read write control password is changed to a read without password control password all secondary

passwords will automatically become read without password secondary passwords.

If the control password has read without password protection, its secondary passwords must also have read without password protection. A request for a read only or for a read write secondary password will result in a read without password secondary password. If a read without password control password is changed to either a read only or read write control password all its secondary passwords will automatically become read write secondary passwords.

PROGRAMMING CONVENTIONS FOR THE PROTECT MACRO INSTRUCTION

The format of the PROTECT macro is:

Name	Operation	Operand
[symbol] (optional)	PROTECT	{(1) register 1: address of a parameter list (REG) any register: address of a parameter list list addr address of the parameter list}

When you issue the PROTECT macro, you should have already established the parameter list. Its size and contents depend on the function that you want the macro to perform. In any case, the first byte of the parameter list is an entry code that indicates the function:

- X'01' for adding an entry to the parameter list.
- X'02' for replacing an entry in the parameter list.
- X'03' for deleting an entry from the parameter list.
- X'04' for listing the information in a password data set entry. For a complete discussion of the contents of the parameter lists, see figures PSWD1 to PSWD4 and the notes explaining each of these figures.

PROTECT Macro Parameter Lists

The parameter lists, their formats and contents are:

0 X'01'	1 00 00 00
4 Data Set Length	5 Pointer to Data Set Name
8 00	9 00 00 00
12 00	13 Pointer to Control Password
16 Number of Volumes	17 Pointer to Volume List
20 Protection Code	21 Pointer to New Password
24 String Length	25 Pointer to String

Figure PSWD1. Parameter List for Add Function

Explanatory Notes for Figure PSWD1.

0 X'01'
Entry code indicating add function.

13 Pointer to control password.
The control password is the password assigned when the data set was placed under protection for the first time. The pointer can be three bytes of binary zeros if the new password is the control password.

16 Number of volumes.
If the data set is not cataloged and you want to have it flagged as protected, you have to specify the number of volumes in this field. A zero indicates that the catalog information should be used.

17 Pointer to volume list.
If the data set is not cataloged and you want to have it flagged as protected, you provide the address of a list of volume serial numbers in this field. Zeros indicate that the catalog information should be used.

20 Protection code.
A one-byte number indicating the type of protection: X'00' indicates default protection (for the add function, the default protection is the type of protection specified in the control password record of the data set), X'01' indicates that the data set is to be read and written, X'02' indicates that the data set is to be read only and X'03' indicates that the data set can be read without a password, but a password is needed to write into it. The PROTECT macro will use the protection code value, specified in the parameter list, to set the protection mode indicator in the password record.

21 Pointer to new password.
If the data set is being placed under protection for the first time, the new password becomes the control password. If you are adding a secondary entry, the new password is different from the control password.

24 String length.
The length of the character string (maximum 77 bytes) that you want to place in the optional information field of the password record. If you do not want to add information, set this field to zero.

25 Pointer to string.
The address of the character string that is going to be put in the optional information field. If you do not want to add additional information, set this field to zero.

0 X'02'	1 00 00 00
4 Data Set Length	5 Pointer to Data Set Name
8 00	9 Pointer to Current Password
12 00	13 Pointer to Control Password
16 Number of Volumes	17 Pointer to Volume List
20 Protection Code	21 Pointer to New Password
24 String Length	25 Pointer to String

Figure PSWD2. Parameter List for Replace Function

Explanatory Notes for Figure PSWD2.

0 X'02'
Entry code indicating REPLACE function

9 Pointer to current password.
The address of the password that is going to be replaced.

13 Pointer to control password.
The address of the password assigned to the data set when it was first placed under protection. The pointer can be two bytes of binary zeros if the current password is the control password.

16 Number of volumes.
If the data set is not cataloged and you want to have it flagged as protected, you have to specify the number of volumes in this field. A zero indicates that the catalog information should be used.

17 Pointer to volume list.
If the data set is not cataloged and you want to have it flagged as protected, you have to provide the address of a list of volume serial numbers in this field. If this field is zero, the catalog information will be used.

20 Protection code.
A one-byte number indicating the type of protection: X'00' indicates that the protection is default protection (for the replacefunction the default protection is the protection specified in the current password record of the data set), X'01' indicates that the data set is to be read and written, X'02' indicates that the data set is to be read only, and X'03' indicates that the data set can be read without a password, but a password is needed to write into the data set.

21 Pointer to new password.
The address of the password that you want to replace the current password.

24 String length.
The length of the character string (maximum 77 bytes) that you want to place in the optional information field of the password record. Set to zero if you do not want to add additional information.

25 Pointer to string.
The address of the character string that is going to be put in the optional information field of the password record. Set the address to zero if you do not want to add additional information.

0 X'03'	1 00 00 00
4 Data Set Length	5 Pointer to Data Set Name
8 00	9 Pointer to Current Password
12 00	13 Pointer to Control Password
16 Number of Volumes	17 Pointer to Volume List

Figure PSWD3. Parameter List for Delete Function

Explanatory Notes for Figure PSWD3.

0 X'03'.
Entry code indicating delete function.

9 Pointer to current password.
The address of the password that you want to delete. You can delete either a control entry or a secondary entry.

13 Pointer to control password.
The address of the password assigned to the data set when it was placed under protection for the first time. This can be zeros if the current password is also the control password.

16 Number of volumes.
If the data set is not cataloged and you want to have it flagged as protected, you have to specify the number of volumes in this field. A zero indicates that the catalog information should be used.

17 Pointer to volume list.
If the data set is not cataloged and you want to have it flagged as protected, you have to provide the address of a list of volume serial numbers in this field. If this field is zero, the catalog information will be used.

0 X'04'	1 Address of 80 Byte Buffer
4 Data Set Length	5 Address of Data Set Name
8 00	9 Address of Current Password

Figure PSWD4. Parameter List for List Function

Explanatory notes for using Figure PSWD4.

0 X'04'.
Entry code indicating list function.

1 Address of 80-byte buffer.
The address of a buffer where the list of information can be returned to your program by the macro instruction.

9 Pointer to current password.
The address of the password of the record that you want listed.

Return Codes from the PROTECT Macro

When the PROTECT macro finished processing, register 15 will contain a return code that indicates what happened during the processing. Figure PSWD5 contains the return codes and their explanations.

Register 15	Explanation
0	The updating of the password data set was successfully completed.
4	The password of the data set name was already in the password data set.
8	The password of the data set name was not in the password data set.
12	A control password is required or the one supplied is incorrect.
16	The supplied parameter list was incomplete or incorrect.
20	There was an I/O error in the password data set.
**24	The password data set was full.
28	The validity check of the buffer address failed.
*32	The LOCATE macro failed. LOCATE's return code is in register 1 and the number of indexes searched is in register 0.
*36	The OBTAIN macro failed. OBTAIN's return code is in register 1.
*40	The DSCB could not be updated.
44	The password data set does not exist.
*48	Tape data set can not be protected.
*52	Data set in use.

*For these return codes, the password data set has been updated, but the DSCB has not been flagged to indicate the protected status of the data set.
**For this return code, a message is written to the console indicating that the password data set is full.

Figure PSWD5. Return Codes from the PROTECT Macro

System Macro Instructions

This chapter contains the description and formats of macro instructions that allow you either to modify control blocks or to obtain information from control blocks and system tables. Before reading this chapter, you should be familiar with the information contained in the prerequisite publications listed below.

PREREQUISITE PUBLICATIONS

The IBM System/360 Operating System: Assembler Language publication (GC28-6514) contains the information necessary to code programs in the assembler language.

The IBM System/360 Operating System: System Control Blocks publication (GC28-6628) contains format and field descriptions of the system control blocks referred to in this chapter.

SMI

SYSTEM MACRO INSTRUCTIONS IN THIS PUBLICATION

The following system macro instructions are described in the chapters of this publication that deal with the subjects shown.

Macro Instruction	Chapter Subject
ATLAS	EXCP Macro Instruction
CAMLST	VTOC Maintenance
CATALOG	Catalog Maintenance
CLOSE	EXCP Macro Instruction
CVT	XDAP Macro Instruction (Appendix)
DCB	EXCP Macro Instructions
DEVTYPE	System Macro Instructions
EOV	EXCP Macro Instruction
EXCP	EXCP Macro Instruction
IECDSECT, IEFJFCBN, IEFUCBOB	IECDSECT, IEFJFCBN, IEFUCBOB Macro Instructions
INDEX	Catalog Maintenance
LABEL	System Macro Instructions
LOCATE	Catalog Maintenance
OBTAIN	Catalog Maintenance
OPEN	EXCP Macro Instruction
...,TYPE=J	System Macro Instructions (Read a JFCB)
POST, WAIT (ECB)	EXCP Macro Instruction, XDAP Macro Instruction
PURGE	EXCP Macro Instruction (Appendix)
RDJFCB	System Macro Instructions
RENAME	VTOC Maintenance, Data Set Protection
RESTORE	EXCP Macro Instruction (Appendix)
SCRATCH	VTOC Maintenance, Data Set Protection
XDAP	XDAP Macro Instruction

Locate Device Characteristics (DEVTYPE) Macro Instruction

The DEVTYPE macro instruction is used to request information relating to the characteristics of an I/O device, and to cause this information to be placed into a specified area. (The results of a DEVTYPE macro instruction executed before a checkpoint is taken should not be considered valid after a checkpoint/restart occurs.)

Name	Operation	Operand
[symbol]	DEVTYPE	ddloc-addrx,area-addrx[,DEVTAB][,RPS]

ddloc-addrx
: specifies the address of a doubleword that contains the symbolic name of the DD statement to which the device is assigned. The name must be left justified in the doubleword, and must be followed by blanks if the name is less than eight characters. The doubleword need not be on a doubleword boundary.

area-addrx
: specifies the address of an area into which the device information is to be placed. The area can be two, five, or six full words, depending on whether or not the DEVTAB and RPS operands are specified. The area must be on a fullword boundary.

DEVTAB
: If DEVTAB is specified, and the device is a direct access device, five full words of information are placed into your area. If DEVTAB is specified, and the device is not a direct access device, two fullwords of information are placed into your area. If DEVTAB is not specified, two fullwords of information are placed into your area.

RPS
: If RPS is specified, DEVTAB must also be specified. The RPS parameter causes one additional full word of RPS information to be included with the DEVTAB information.

Note: Any reference to a dummy DD statement in the DEVTYPE macro instruction will cause zeroes to be placed in the output area.

Device Characteristics Information

The following information is placed into your area:

Word 1 (offset 0)	Device code from the UCB in which:		
	Byte 1	bit 0 Unassigned	
		bit 1 Overrunable device	1 = yes
		bit 2 Burst/byte mode	1 = burst
		bit 3 Data chaining	1 = yes
		bit 4-7 Model code	
	Byte 2	Optional features	
	Byte 3	Device classes	
	Byte 4	Unit type	

Note: Bit settings for byte 2 are noted in the UCB format and field description in the System Control Blocks publication.

Word 2 (offset 1)		Maximum block size. For direct access devices, this value is the maximum size of an unkeyed block; for magnetic or paper tape devices, this value is the maximum block size allowed by the operating system. For all other devices, this value is the maximum block size accepted by the device.

If DEVTAB is specified, the next three full words contain the following information:

Word 3 (offset 2)	Bytes 1-2	The number of physical cylinders on the device.
	Bytes 3-4	The number of tracks per cylinder.
Word 4 (offset 3)	Bytes 1-2	Maximum track length. Note that for the 2305 and 3330 direct access devices this value is not equal to the value in word two (maximum block size) as it is for other IBM direct access devices.
	Byte 3	Block overhead - the number of bytes required for gaps and check bits for each keyed block other than the last block on a track.
	Byte 4	Block overhead - the number of bytes required for gaps and check bits for a keyed block that is the last block on a track.
	Bytes 3-4	Block overhead - the number of bytes required for gaps and check bits for any keyed block on a track including the last block. Use of this form is indicated by a one in bit 4, byte 2 of word 5.
Word 5 (offset 4)	Byte 1	Block overhead - the number of bytes to be subtracted if a block is not keyed.
	Byte 2	bits 0-3 Reserved. bit 4 If 1, bytes 3 and 4 of word 4 contain a halfword giving the block overhead for any block on a track, including the last bit 5 Reserved bit 6 Unconditionally on for 2321; off for others bit 7 If 1, a tolerance factor must be applied to all blocks except the last block on the track.
	Bytes 3-4	Tolerance factor - this factor is used to calculate the effective length of a block. The calculation should be performed as follows: Step 1 - add the block's key length to the block's data length. Step 2 - test bit 7 of byte 2 of word 5. If bit 7 is 0, perform step 3. If bit 7 is 1, multiply the sum computed in step 1 by the tolerance factor. Shift the result of the multiplication nine bits to the right. Step 3 - add the appropriate block overhead to the value obtained above.

If DEVTAB and RPS are specified, the next full word contains the following information:

Word 6 (offset 5)	Bytes 1-2	R0 overhead for sector calculations.
	Byte 3	Number of sectors for the device.
	Byte 4	Number of data sectors for the device.

Output for Each Device Type

	UCB Type Field (Word 1, In Hexadecimal)	Maximum Record Size (Word 2, In Decimal)	DEVTAB (Words 3, 4, and 5, In Hexadecimal)	RPS (Word 6 In Hexadecimal)
2540 Reader	10 00 08 01	80	Not Applicable	Not Applicable
2540 Reader w/CI	10 01 08 01	80	Not Applicable	Not Applicable
2540 Punch	10 00 08 02	80	Not Applicable	Not Applicable
2540 Punch w/CI	10 01 08 02	80	Not Applicable	Not Applicable
1442 Reader-Punch	50 00 08 03	80	Not Applicable	Not Applicable
1442 Reader-Punch w/CI	50 01 08 03	80	Not Applicable	Not Applicable
1442 Serial Punch	51 80 08 03	80	Not Applicable	Not Applicable
1442 Serial Punch w/CI	51 01 08 03	80	Not Applicable	Not Applicable
2501 Reader	50 00 08 04	80	Not Applicable	Not Applicable
2501 Reader w/CI	50 01 08 04	80	Not Applicable	Not Applicable
2520 Reader Punch	50 00 08 05	80	Not Applicable	Not Applicable
2520 Reader Punch w/CI	50 01 08 05	80	Not Applicable	Not Applicable
2520 B2-B3	11 00 08 05	80	Not Applicable	Not Applicable
2520 B2-B3 w/CI	11 01 08 05	80	Not Applicable	Not Applicable
1403	10 00 08 08	120*	Not Applicable	Not Applicable
1403 w/UCS	10 80 08 08	120*	Not Applicable	Not Applicable
1404	10 00 08 08	120*	Not Applicable	Not Applicable
1443	10 00 08 0A	120*	Not Applicable	Not Applicable
3211	10 80 08 09	132*	Not Applicable	Not Applicable
2671	10 00 08 10	32767	Not Applicable	Not Applicable
1052	10 00 08 20	130	Not Applicable	Not Applicable
2400 (9-track)	30 00 80 01	32767	Not Applicable	Not Applicable
2400 (9-track, p.e.)	34 00 80 01	32767	Not Applicable	Not Applicable
2400 (9-track, d.d.)	34 20 80 01	32767	Not Applicable	Not Applicable
2400 (7-track)	30 80 80 01	32767	Not Applicable	Not Applicable
2400 (7-track, d.c.)	30 C0 80 01	32767	Not Applicable	Not Applicable
2301	30 40 20 02	20483	000100C85003BA3535000200	Not Applicable
2302	30 00 20 04	4984	00FA002E1378511414010219	Not Applicable
2303	30 00 20 03	4892	0050000A131C922626000200	Not Applicable
2311	30 00 20 01	3625	00CB000A0E29511414010219	Not Applicable
2314	30 C0 20 08	7294	00CB00141C7E922D2D010216	Not Applicable
2321	30 00 20 05	2000	140A051407D0641010030219	Not Applicable
3210 Printer Keyboard	10 00 08 22	130	Not Applicable	Not Applicable
3215 Printer Keyboard	10 00 08 23	130	Not Applicable	Not Applicable
3505 Card Reader	10 00 08 06	80	Not Applicable	Not Applicable
3505 Card Reader w/CI	10 01 08 06	80/160	Not Applicable	Not Applicable
3525 Card Punch	10 00 08 0C	80	Not Applicable	Not Applicable
3525 Card Punch w/CI	10 01 08 0C	80/160	Not Applicable	Not Applicable
2305-1	30 50 20 06	14,138	0030000838E80278CA090200	02985A57
2305-2	30 50 20 07	14,660	006A00083A0A01215B090200	0140B4B1
3330	30 50 20 09	13,030	019B0013336DBFBF38010200	00ED807C

Device	UCB Type Field (Word 1, In Hexadecimal)	Maximum Record Size (Word 2, In Decimal)	DEVTAB (Words 3, 4, and 5, In Hexadecimal)	RPS (Word 6 In Hexadecim
1053	14 00 10 04		Not Applicable	Not Applicabl
2250 (Mod 1)	31 xx 10 02		Not Applicable	Not Applicabl
2250 (Mod 2)	32 xx 10 02		Not Applicable	Not Applicabl
2250 (Mod 3)	33 xx 10 02		Not Applicable	Not Applicabl
2280	30 00 10 05		Not Applicable	Not Applicabl
2282	30 00 10 06		Not Applicable	Not Applicabl
3066 (Model 165 System Console)	10 00 10 08		Not Applicable	Not Applicabl
5450 (Model 85 Operators Console)	10 00 10 07		Not Applicable	Not Applicabl
3400 (9-track,p.e.)	34 00 80 03	32767	Not Applicable	Not Applicabl
3400 (9-track, d.d.)	34 20 80 03	32767	Not Applicable	Not Applicabl
3400 (7-track)	30 C0 80 03	32767	Not Applicable	Not Applicabl

Legend

CI-Card image feature, d.c.-data conversion, d.d.-dual density, p.e.-phase encoding, UCS-universal character set, w/-with

*Although certain models can have a larger line size, the minimum line size is assumed.

xx = Special feature (byte 2) configurations may be obtained from the System Control Blocks publication.

Communication Equipment	UCB Type Field	Record Size
1030,1050,83B3, TWX,2250, S360	51xx40YZ	Not Applicable
1060,115A,1130	52xx40YZ	Not Applicable
2780	53xx40YZ	Not Applicable
2740	54xx40YZ	Not Applicable

Y=Adapter Type (Bits 0-3)

Hex Value	Meaning
1	IBM Terminal Adapter, Type I
2	IBM Terminal Adapter, Type II
3	IBM Telegraph Adapter
4	Telegraph Adapter, Type I
5	Telegraph Adapter, Type II
6	World Trade Telegraph Adapter
7	Synchronous Adapter, Type I
8	IBM Terminal Adapter, Type III
9	Synchronous Adapter, Type II

Z=Control Unit (Bits 4-7)

Hex value	Meaning
1	2702
2	2701
3	2703

Exceptional Returns

The following return codes are placed in register 15:

00 - Request completed satisfactorily.
04 - Ddname not found.
08 - Invalid area address. The address of the output area either violates protection, or it is out of the range of main storage.

How to Read a Job File Control Block

To accomplish the functions that are performed as a result of an OPEN macro instruction, the OPEN routine requires access to information that you have supplied in a data definition (DD) statement. This information is stored by the system in a job file control block (JFCB).

Usually, the programmer is not concerned with the JFCB itself. In special applications, however, you may find it necessary to modify the contents of a JFCB before issuing an OPEN macro instruction. To assist you, the system provides the RDJFCB macro instruction. This macro instruction causes a specified JFCB to be read into main storage from the job queue in which it has been stored. Format and field description of the JFCB is contained in the System Control Blocks publication.

When subsequently issuing the OPEN macro instruction, you must indicate, by specifying the TYPE=J option, that you have supplied a modified JFCB to be used during the initialization process.

The JFCB is returned to the job queue by the OPEN routine or the OPENJ routine, if any of the modifications in the following list occur. These modifications can occur only if the information is not originally in the JFCB.

- Expiration date field and creation date field merged into the JFCB from the DSCB.
- Secondary quantity field merged into the JFCB from the DSCB.
- DCB fields merged into the JFCB from the DSCB.
- DCB fields merged into the JFCB from the DCB.
- Volume serial number fields added to the JFCB.
- Data set sequence number field added to the JFCB.
- Number of volumes field added to the JFCB.

If you make these, or any other modifications, and you want the JFCB returned to the job queue, you must set the high-order bit of field JFCBMASK+4 to one. This field is in the JFCB. Setting the high-order bit of field JFCBMASK+4 to zero does not necessarily suppress the return of the JFCB to the job queue. If the OPEN or OPENJ routines have made any of the above modifications, the JFCB is returned to the job queue. To inhibit writing the JFCB back to the job queue during an OPENJ, the field JFCBTSDM should be set to X'08' prior to issuing the OPEN macro.

OPEN -- PREPARE THE DATA CONTROL BLOCK FOR PROCESSING (S)

The OPEN macro instruction initializes one or more data control blocks so that their associated data sets can be processed.

A full explanation of the operands of the OPEN macro instruction, except for the TYPE=J option, is contained in the Data Management Macro Instructions publication. The TYPE=J option, because it is used in conjunction with modifying a JFCB, should be used only by the system programmer or only under his supervision.

Name	Operation	Operand
[symbol]	OPEN	({dcb-addr,[(opt_1-code[,opt_2-code])],}...) [,TYPE=J]

TYPE=J
specifies that, for each data control block referred to, the programmer has supplied a job file control block (JFCB) to be used during initialization. A JFCB is an internal representation of information in a DD control statement.

During initialization of a data control block, its associated JFCB may be modified with information from the data control block or an existing data set label or with system control information.

The system always creates a job file control block for each DD control statement. The job file control block is placed in a job queue on direct access storage. Its position, in relation to other JFCBs created for the same job step, is noted in a main storage table.

When this operand is specified, the user must also supply a DD control statement. However, the amount of information given in the DD statement is at the programmer's discretion, because he can ignore the system-created job file control block. (See the examples of the RDJFCB macro instruction for a technique for modification of a system-created JFCB.)

Caution: In MVT configurations of the operating system, data set integrity provided by the job scheduler functions is lost if you change, or do not use, the DSNAME=parameter in the DD statement.

Note: The DD statement must specify at least:

- Device allocation (refer to the Job Control Language publication for methods of preventing share status).
- A ddname corresponding to the associated data control block DCBDDNAM field.

RDJFCB -- READ A JOB FILE CONTROL BLOCK (S)

The RDJFCB macro instruction causes a job file control block (JFCB) to be read from the job queue into main storage for each data control block specified.

Name	Operation	Operand
[symbol]	RDJFCB	({dcb-addr,[(opt_1-code[,opt_2-code])],}...)

dcb,(opt_1,opt_2)
(same as dcb, opt_1, and opt_2 operands in OPEN macro instruction)

Although the opt_1 and opt_2 operands are not meaningful during the execution of the RDJFCB macro instruction, these operands can appear in the L-form of either the RDJFCB or OPEN macro instruction to generate identical parameter lists, which can be referred to with the E-form of either macro instruction.

Examples: The macro instruction in EX1 creates a parameter list for two data control blocks: INVEN and MASTER. In creating the list, both data control blocks are assumed to be opened for input; opt_2 for both blocks is assumed to be DISP. The macro instruction in EX2 reads the system-created JFCBs for INVEN and MASTER from the job queue into main storage, thus making the JFCBs available to the problem program for modification. The macro instruction in EX3 modifies the parameter list entry for the data control block named INVEN and indicates, through the TYPE=J operand, that the problem program is supplying the JFCBs for system use.

```
EX1     RDJFCB (INVEN,,MASTER),MF=L
        .
        .
EX2     RDJFCB MF=(E,EX1)
        .
        .
EX3     OPEN   (,(RDBACK,LEAVE)),TYPE=J,MF=(E,EX1)
        .
        .
```

Programming Notes

Any number of data control block addresses and associated options may be specified in the RDJFCB macro instruction. This facility makes it possible to read job file control blocks in parallel.

An exit list address must be provided in each data control block specified by an RDJFCB macro instruction. Each exit list must contain an active entry that specifies the main storage address of the area into which a JFCB is to be placed. A full discussion of the exit list and its use is contained in the Data Management Services publication. The format of the job file control block exit list entry is as follows:

Type of Exit List Entry	Hexadecimal Code (high-order byte)	Contents of Exit List Entry (three low-order bytes)
Job file control block	07	Address of a 176-byte area to be provided if the RDJFCB or OPEN (TYPE=J) macro instruction is used. This area must begin on a fullword boundary and must be located within the user's region.

The main storage area into which the JFCB is read must be at least 176 bytes long.

The data control block may be open or closed when this macro instruction is executed.

If the JFCB is read successfully for all DCBs in the parameter list, a return code of zero is placed in register 15. If the JFCB is not read for any of the DCBs because the DDNAME is blank or a DD statement is not provided, then a return code of 4 is placed in register 15.

Cautions: The following errors cause the results indicated:

Error	Result
A DD control statement has not been provided.	A return code of 4 is placed in register 15.
DDNAME field in DCB is blank	A write-to-programmer is issued, the request for this DCB is ignored, and a return code of 4 is placed in register 15.
A main storage address has not been provided.	Abnormal termination of task

Adding a Universal Character Set Image or a Forms Control Buffer Image to the Image Library

This chapter provides a detailed description of how to add either an IBM UCS (universal character set) image or an IBM FCB (forms control buffer) image to SYS1.IMAGELIB.

Before reading this section, you should be familiar with the information contained in the publications listed below.

REFERENCE PUBLICATIONS

IBM 2821 Control Unit, GA24-3312, contains the information necessary to create a user-designed chain/train for the 1403 Printer.

OS Data Management Macro Instructions, GC26-3794, describes the SETPRT macro instruction that loads a UCS image and an FCB image into their respective buffers.

OS Job Control Language Reference, GC28-6704, describes the UCS and FCB parameters that can be specified in a DD statement to load the UCS and FCB buffers when they are opened.

IBM 3211 Printer and 3811 Control Unit Component Description, GA24-3543, contains the information necessary to create a user-designed train for the 3211 Printer.

UCS/ FCB

How to Add a UCS Image to the Image Library

The IBM standard character set images listed in the following table may be included in SYS1.IMAGELIB at system generation by using the UCS macro instruction. You code a member name for an image in the image library by prefixing a character set code shown in the table with UCS1 or UCS2: UCS1 denotes a 1403 printer image and UCS2 denotes a 3211 printer image (for example, UCS1AN or UCS2A11).

1403	AN, HN, PCAN, PCHN, PN, QNC, QN, RN, SN, TN, XN, YN
3211	A11, G11, H11, P11, T11

You may add a user-designed character image to the image library or make an existing image a default image by following these rules:

1. The member name must be either the four characters UCS1 for the 1403 or UCS2 for the 3211 printer. The member name must be followed by a unique character set code that is one to four characters long. This character set code can be any valid combination of letters and numbers according to the rules for assembler language symbols. The single letters U or C should not be used as a character set code since they are symbols for special conditions recognized by the system. The assigned character set code must be specified on the DD statement or SETPRT macro instruction to load the image into the UCS buffer.

2. The first byte in the load module of a character set image specifies whether or not the image is a default. A default image is indicated by X'80', and is used when the UCS parameter is not coded in the DD statement. X'00' specifies that the image is not to be used as a default.

3. The second byte of the load module indicates the number of lines (n) to be printed for image verification.

4. Each byte if the next n bytes indicates the number of characters to be printed on each verification line. (Note: For the 3211 printer, the maximum number of characters printed per line is 48; the associative bytes are not printed during verification.)

5. A 240-byte 1403 UCS image or a 512-byte 3211 UCS image must follow the previously described fields. (A 3211 UCS image has 432 characters, followed by 15 bytes of X'00', 64 bytes of associative bits, and a reserved byte (byte 512) of X'00'.) Two apostrophes or two ampersands must be coded to represent a single apostrophe or a single ampersand, respectively, that is a part of a character set image.

The following code is an example of adding a 1403 UCS image, YN, to the image library.

```
//ADDYN     JOB  MSGLEVEL=1
//STEP      EXEC PROC=ASMFCL,PARM.ASM='NODECK,LOAD',                X
//               PARM.LKED='LIST,NCAL,NE,OL'
//ASM.SYSIN DD   *
UCS1YN      CSECT
            DC   X'80'        (this is a default image)
            DC   AL1(6)       (number of lines to be printed)
            DC   AL1(39)      (39 characters printed on 1st line)
            DC   AL1(42)      (42 characters printed on 2nd line)
            DC   AL1(39)      (39 characters printed on 3rd line)
            DC   AL1(39)      (39 characters printed on 4th line)
            DC   AL1(42)      (42 characters printed on 5th line)
            DC   AL1(39)      (39 characters printed on 6th line)
            DC   C'1234567890STABCDEFGHIJKLMNOPQRSTUVWXYZ*,.'
            DC   C'1234567890STABCDEFGHIJKLMNOPQRSTUVWXYZ*,.#-$'
            DC   C'1234567890STABCDEFGHIJKLMNOPQRSTUVWXYZ*,.'
            DC   C'1234567890STABCDEFGHIJKLMNOPQRSTUVWXYZ*,.'
            DC   C'1234567890STABCDEFGHIJKLMNOPQRSTUVWXYZ*,.#-$'
            DC   C'1234567890STABCDEFGHIJKLMNOPQRSTUVWXYZ*,.'
            END
/*
//LKED.SYSLMOD DD DSNAME=SYS1.IMAGELIB(UCS1YN),DISP=OLD
```

The following example shows the code used to add a 3211 UCS image (A11) to the image library. A 3211 UCS image has 432 characters, followed by 15 bytes of X'00', 64 bytes of associative bits, and a reserved byte (byte 512) of X'00'. Two ampersands must be coded to represent a single ampersand that is part of the character set image.

The 64 bytes of associative bits must be coded to avoid data checks. To determine how to code these bits for a particular chain, see the IBM 3211 Printer, 3216 interchangeable Train Cartridge, and 3811 Printer Control Unit Component Description and Operator's Guide, GA24-3543.

```
//ADDA11      JOB   MSGLEVEL=1
//STEP        EXEC  PROC=ASMFCL,PARM.ASM='NODECK,LOAD',                       X
//                  PARM.LKED='LIST,NCAL,NE,OL'
//ASM.SYSIN   DD    *
UCS2A11       CSECT
              DC    X'80'        (THIS IS A DEFAULT IMAGE)
              DC    AL1(9)       (NUMBER OF LINES TO BE PRINTED)
              DC    AL1(48)      (48 CHARACTERS PRINTED ON 1ST LINE)
              DC    AL1(48)      (48 CHARACTERS PRINTED ON 2ND LINE)
              DC    AL1(48)      (48 CHARACTERS PRINTED ON 3RD LINE)
              DC    AL1(48)      (48 CHARACTERS PRINTED ON 4TH LINE)
              DC    AL1(48)      (48 CHARACTERS PRINTED ON 5TH LINE)
              DC    AL1(48)      (48 CHARACTERS PRINTED ON 6TH LINE)
              DC    AL1(48)      (48 CHARACTERS PRINTED ON 7TH LINE)
              DC    AL1(48)      (48 CHARACTERS PRINTED ON 8TH LINE)
              DC    AL1(48)      (48 CHARACTERS PRINTED ON 9TH LINE)
*THE FOLLOWING NINE LINES REPRESENT THE TRAIN IMAGE
              DC    C'1<.+IHGFEDCBA*$-RQPONMLKJ%,&&ZYXWVUTS/@#098765432'
              DC    C'1<.+IHGFEDCBA*$-RQPONMLKJ%,&&ZYXWVUTS/@#098765432'
              DC    C'1<.+IHGFEDCBA*$-RQPONMLKJ%,&&ZYXWVUTS/@#098765432'
              DC    C'1<.+IHGFEDCBA*$-RQPONMLKJ%,&&ZYXWVUTS/@#098765432'
              DC    C'1<.+IHGFEDCBA*$-RQPONMLKJ%,&&ZYXWVUTS/@#098765432'
              DC    C'1<.+IHGFEDCBA*$-RQPONMLKJ%,&&ZYXWVUTS/@#098765432'
              DC    C'1<.+IHGFEDCBA*$-RQPONMLKJ%,&&ZYXWVUTS/@#098765432'
              DC    C'1<.+IHGFEDCBA*$-RQPONMLKJ%,&&ZYXWVUTS/@#098765432'
              DC    C'1<.+IHGFEDCBA*$-RQPONMLKJ%,&&ZYXWVUTS/@#098765432'
              DC    15X'00'      RESERVED FIELD, BYTES 433-447
*THE FOLLOWING FOUR DC INSTRUCTIONS DEFINE THE ASSOCIATIVE BITS,
*UCSB BYTE POSITIONS 448-511
              DC    X'C0101010101010101010000404042400004010'
              DC    X'101010101010101000040404100004040101O'
              DC    X'10101010101000404000000010101010101O'
              DC    X'10101010004040444800'
              DC    X'00'        RESERVED BYTE, BYTE 512
              END
/*
//LKED.SYSLMOD  DD  DSNAME=SYS1.IMAGELIB(UCS2A11),DISP=OLD
```

Note: Executing the ASMFCL procedure does not actually generate executable code. The assembler/linkage editor is used as a vehicle to load the UCS image into the image library.

How to Add a Forms Control Buffer Image to the Image Library

Two standard FCB images, STD1 and STD2, can be included in SYS1.IMAGELIB during system generation for a 3211 printer. STD1 prints six lines per inch on a 8 1/2 inch form. STD2 prints six lines per inch on an eleven inch form. Channels for both images are evenly spaced with channel one on the fourth line and channel nine on the last line.

In addition to the IBM-supplied images, user images can be defined. Each user image is added to the image library as part of a load module. To add an FCB image to the image library, follow these rules:

1. The member name cannot exceed eight bytes. The first four characters of this member name must be FCB2. The characters that follow FCB2 identify the FCB image and are referred to as the image identifier. Any combination of characters that are valid in assembly language can be used with the exception of a single "S" or a single "U" as an image identifier. The image identifier must be specified on a DD statement or in the SETPRT macro instruction to load the image in the FCB buffer.

2. The first byte of the load module of a forms control image specifies whether or not the image is a default. A default image is indicated by X'80' and is used for all jobs that do not have the FCB parameter coded on the DD statement; X'00' indicates that the image is not to be used as a default.

3. The second byte of the load module indicates the number of lines per form (FCB image length). The maximum image length is 180 lines. The FCB image must be as long as the form. For example, if you are printing eight lines per inch on an eleven inch form, the FCB image must be 88 bytes long; if you are printing six lines per inch on the same form, the FCB image must be 66 bytes long.

4. The first of the FCB image (the third byte of the load module) defines the number of lines per inch and a channel:

   - X'1n' means eight lines are printed per inch.
   - X'0n' means six lines are printed per inch.

   All remaining bytes (lines) must contain X'0n' except the last byte. The last byte must be X'1n'. The letter n can be a hexadecimal value from 1 to C representing a channel (one to twelve); or it can be zero (0) which means no channel is indicated.

In the following example, an FCB load module is assembled and added to SYS1.IMAGELIB. The image defines a print density of eight lines per inch on an eleven inch form.

```
//ADDFCB    JOB  MSGLEVEL=1
//STEP      EXEC PROC=ASMFCB,PARM.ASM='NODECK,LOAD',                    X
//             PARM.LKED='LIST,NCAL,NE,OL'
//ASM.SYSIN DD   *
FCB2ID1     CSECT
*THIS EXAMPLE IS FOR A FORM LENGTH OF 11 INCHES
*WITH 8 LINES OF PRINT PER INCH (88 LINES)
            DC   X'80'       THIS IS A DEFAULT IMAGE
            DC   AL1(88)     LENGTH OF FCB IMAGE
            DC   X'10'       8 LINES PER INCH-NO CHANNEL FOR POS.1
            DC   XL4'0'      4 LINES NO CHANNEL
            DC   X'01'       CHANNEL 1 IN POSITION 6
            DC   XL6'0'      6 LINES NO CHANNEL
            DC   X'02'       CHANNEL 2 IN POSITION 13
            DC   XL6'0'
            DC   X'03'
            DC   XL6'0'
            DC   X'04'
            DC   XL6'0'
            DC   X'05'
            DC   XL6'0'
            DC   X'06'
            DC   XL6'0'
            DC   X'07'
            DC   XL6'0'
            DC   X'08'
            DC   XL6'0'
            DC   X'09'
            DC   XL6'0'
            DC   X'0A'
            DC   XL6'0'
            DC   X'0B'
            DC   XL6'0'
            DC   X'0C'       CHANNEL 12 IN POSITION 83
            DC   XL4'0'      4 LINES NO CHANNEL
            DC   X'10'       POSITION 88 - LAST LINE IN IMAGE
            END
/*
//LKED.SYSLMOD  DD  DSNAME=SYS1.IMAGELIB(FCB2ID1),DISP=OLD
```

OS Data Management for System Programmers (File No. S360-30) Printed in U.S.A. GC28-6550-11

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)

OS Data Management for System Programmers

GC28-6550-11

Reader's Comment Form

Your comments about this publication will help us to improve it for you. Comment in the space below, giving specific page and paragraph references whenever possible. All comments become the property of IBM.

Please do not use this form to ask technical questions about IBM systems and programs or to request copies of publications. Rather, direct such questions or requests to your local IBM representative.

If you would like a reply, please provide your name, job title, and business address (including ZIP code).

Fold on two lines, staple, and mail. No postage necessary if mailed in the U.S.A. (Elsewhere, any IBM representative will be happy to forward your comments.) Thank you for your cooperation.

GC28-6550-11

Fold and Staple

First Class Permit
Number 2078
San Jose, California

Business Reply Mail
No postage necessary if mailed in the U.S.A.

Postage will be paid by:

IBM Corporation
Programming Center – Publishing
Department D58
Monterey and Cottle Roads
San Jose, California 95193

Fold and Staple

OS Data Management for System Programmers (File No. S360-30) Printed in U.S.A. GC28-6550-11

IBM

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)