GC26-4065-1
File No. S370-32

Program Product

MVS/370 Utilities

Data Facility Product 5665-295

Release 1.1

Second Edition (October 1983)

This edition, as amended by technical newsletter GN26-8133, applies to Release 1.1 of MVS/370 Data Facility Product, Program Product 5665-295, and to any subsequent releases until otherwise indicated in new editions or technical newsletters.

The changes for this edition are summarized under "Summary of Amendments" following the preface. Specific changes are indicated by a vertical bar to the left of the change. These bars will be deleted at any subsequent republication of the page affected. Editorial changes that have no technical significance are not noted.

Changes are made periodically to this publication; before using this publication in connection with the operation of IBM systems, consult the latest IBM System/370 and 4300 Processors Bibliography, GC20-0001, for the editions that are applicable and current.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM program product in this publication is not intended to state or imply that only IBM's program product may be used. Any functionally equivalent program may be used instead.

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

A form for readers' comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, P.O. Box 50020, Programming Publishing, San Jose, California, U.S.A. 95150. IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

PREFACE

This publication describes how to use the MVS/370 Data Facility Product (MVS/370) utility programs to manipulate system and user data and data sets.

ORGANIZATION

This publication contains the following major parts:

- "Introduction" summarizes the utility programs and information on the differences among system, data set, and independent utility programs. The introduction contains basic information about how the programs are executed and about the utility control statements used to specify program functions. New or infrequent users of the utility programs should give particular attention to the introduction.
- "Guide to Utility Program Functions" contains a table, arranged in alphabetic order, of utility program functions and the programs that perform them. This table enables you to find the program that can do what you need to have done.
- "Invoking Utility Programs from a Problem Program" contains a description of the macro instructions used to invoke a utility program from a problem program rather than executing the utility program by job control statements or by a procedure in the procedure library. This section should be read only if you plan to invoke a utility program from a problem program.
- The remainder of the book contains individual chapters for each utility program arranged in alphabetic order. For a discussion of the organization of these chapters, see "Organization of Program Descriptions" on page iv.
- Appendix A, "Exit Routine Linkage" contains information about linking to and returning from optional user-supplied exit routines. This appendix should be read only if you plan to code or use an exit routine. If you are coding an exit routine, this appendix provides linkage conventions, descriptions of parameter lists, and return codes. If you are using an existing exit routine, you may be interested in the meaning of return codes from the exit routine.
- Appendix B, "DD Statements for Defining Mountable Devices" contains a review of how to define mountable volumes to ensure that no one else has access to them. For a definitive explanation of this subject, see the publication OS/VS2 MVS JCL.
- Appendix C, "Processing User Labels" describes the user-label processing that can be performed by IEBGENER, IEBCOMPR, IEBPTPCH, IEHMOVE, and IEBUPDTE. This appendix should be read only if you plan to use a utility program for processing user labels.

ORGANIZATION OF PROGRAM DESCRIPTIONS

To enable you to find information more easily, program descriptions are all organized, as much as possible, in the same way. Most programs are discussed according to the following pattern:

- Introduction to and description of the functions that can be performed by the program. This description typically includes an overview of the program's use, definitions of terms, illustrations, etc.
- Functions supported by the utility and the purpose of each function.
- Input and output (including return codes) used and produced by the program.
- Control of the program through job control statements and utility control statements. Explanations of utility control statement parameters are presented in alphabetic order in tabular format, showing applicable control statements, syntax, and a description of the parameters. Any general information, restrictions, and relationships of a given utility control statement to other control statements are described in the sections concerning the statements or in the section for restrictions.
- Examples of using the program, including the job control statements and utility control statements.

PREREQUISITE KNOWLEDGE

In order to use this book efficiently, you should be familiar with the following:

- Job control language
- Data management
- Virtual storage management

REQUIRED PUBLICATIONS

You should be familiar with the information presented in the following publications:

- MVS/370 Utilities Messages contains a complete listing and explanation of the messages and codes issued by MVS/370 utility programs.
- OS/VS2 MVS JCL contains a description of the use and coding of the job control language.
- MVS/370 Data Management Services describes the input/output facilities of the operating system. It contains information on record formats, data set organization, access methods, data set disposition, space allocation, and generation data sets.
- MVS/370 Data Management Macro Instructions contains contains a description of the WRITE macro instruction; it also contains the format and contents of the DCB.
- OS/VS2 MVS System Programming Library: Supervisor Services and Macro Instructions contains contains information on how to use the services of the supervisor. Among the services of the supervisor are program management, task creation and management, and virtual storage management.

PREFACE

This publication describes how to use the MVS/370 Data Facility Product (MVS/370) utility programs to manipulate system and user data and data sets.

ORGANIZATION

This publication contains the following major parts:

- "Introduction" summarizes the utility programs and information on the differences among system, data set, and independent utility programs. The introduction contains basic information about how the programs are executed and about the utility control statements used to specify program functions. New or infrequent users of the utility programs should give particular attention to the introduction.
- "Guide to Utility Program Functions" contains a table, arranged in alphabetic order, of utility program functions and the programs that perform them. This table enables you to find the program that can do what you need to have done.
- "Invoking Utility Programs from a Problem Program" contains a description of the macro instructions used to invoke a utility program from a problem program rather than executing the utility program by job control statements or by a procedure in the procedure library. This section should be read only if you plan to invoke a utility program from a problem program.
- The remainder of the book contains individual chapters for each utility program arranged in alphabetic order. For a discussion of the organization of these chapters, see "Organization of Program Descriptions" on page iv.
- Appendix A, "Exit Routine Linkage" contains information about linking to and returning from optional user-supplied exit routines. This appendix should be read only if you plan to code or use an exit routine. If you are coding an exit routine, this appendix provides linkage conventions, descriptions of parameter lists, and return codes. If you are using an existing exit routine, you may be interested in the meaning of return codes from the exit routine.
- Appendix B, "DD Statements for Defining Mountable Devices" contains a review of how to define mountable volumes to ensure that no one else has access to them. For a definitive explanation of this subject, see the publication OS/VS2 MVS JCL.
- Appendix C, "Processing User Labels" describes the user-label processing that can be performed by IEBGENER, IEBCOMPR, IEBPTPCH, IEHMOVE, and IEBUPDTE. This appendix should be read only if you plan to use a utility program for processing user labels.

ORGANIZATION OF PROGRAM DESCRIPTIONS

To enable you to find information more easily, program descriptions are all organized, as much as possible, in the same way. Most programs are discussed according to the following pattern:

- Introduction to and description of the functions that can be performed by the program. This description typically includes an overview of the program's use, definitions of terms, illustrations, etc.
- Functions supported by the utility and the purpose of each function.
- Input and output (including return codes) used and produced by the program.
- Control of the program through job control statements and utility control statements. Explanations of utility control statement parameters are presented in alphabetic order in tabular format, showing applicable control statements, syntax, and a description of the parameters. Any general information, restrictions, and relationships of a given utility control statement to other control statements are described in the sections concerning the statements or in the section for restrictions.
- Examples of using the program, including the job control statements and utility control statements.

PREREQUISITE KNOWLEDGE

In order to use this book efficiently, you should be familiar with the following:

- Job control language
- Data management
- Virtual storage management

REQUIRED PUBLICATIONS

You should be familiar with the information presented in the following publications:

- MVS/370 Utilities Messages contains a complete listing and explanation of the messages and codes issued by MVS/370 utility programs.
- OS/VS2 MVS JCL contains a description of the use and coding of the job control language.
- MVS/370 Data Management Services describes the input/output facilities of the operating system. It contains information on record formats, data set organization, access methods, data set disposition, space allocation, and generation data sets.
- MVS/370 Data Management Macro Instructions contains a description of the WRITE macro instruction; it also contains the format and contents of the DCB.
- OS/VS2 MVS Supervisor Services and Macro Instructions contains information on how to use the services of the supervisor. Among the services of the supervisor are program management, task creation and management, and virtual storage management.

RELATED PUBLICATIONS

Within the text, references are made to the publications listed in the table below.

Short Title	Publication Title	Order Number
Catalog Users Guide	MVS/370 Catalog Users Guide	GC26-4053
Data Facility Data Set Services: User's Guide and Reference	Data Facility Data Set Services: User's Guide and Reference	SC26-3949
Debugging Handbook	OS/VS2 MVS System Programming Library: Debugging Handbook, Volumes 1 through 3	GC28-1047 GC28-1048 GC28-1049
Device Support Facilities User's Guide and Reference	Device Support Facilities User's Guide and Reference	GC35-0033
Linkage Editor and Loader	MVS/370 Linkage Editor and Loader	GC26-4061
Same	IBM 3800 Printing Subsystem Programmer's Guide	GC26-3846
Same	IBM 50 Magnetic Data Inscriber Component Description	GA27-2725
Utilities Messages	MVS/370 Utilities Messages	GC26-4068
JCL	OS/VS2 MVS JCL	GC28-0692
Data Management Macro Instructions	MVS/370 Data Management Macro Instructions	GC26-4057
Supervisor Services and Macro Instructions	OS/VS2 MVS Supervisor Services and Macro Instructions	GC28-0683
Same	Reference Manual for the IBM 3800 Printing Subsystem	GA26-1653
System Programming Library: Data Management	MVS/370 System Programming Library: Data Management	GC26-4056
VSAM Users Guide	MVS/370 VSAM Users Guide	GC26-4066

UTILITIES NOT EXPLAINED IN THIS BOOK

There are several specialized utilities not discussed in this book. The following list shows their names and functions, and indicates which book contains their explanation.

Utility	Function	Reference
IDCAMS	Allows users to define, manipulate, or delete VSAM data sets, define and manipulate VSAM catalogs, and copy, print, or convert SAM and ISAM data sets to VSAM data sets.	MVS/370 Catalog Users Guide, GC26-4053
Device Support Facilities	Used for the initialization and maintenance of DASD volumes.	Device Support Facilities User's Guide and Reference, GC35-0033
Data Facility Data Set Services	Describes DASD utility functions such as dump/restore and reduction of free space fragmentation	Data Facility Data Set Services: User's Guide and Reference, SC26-3949
Offline IBM 3800 Utility	Describes the Offline IBM 3800 Utility program, used with the IBM 3800 Tape-to-Printing Subsystem Feature.	Offline IBM 3800 Utility, SH20-9138

SUMMARY OF AMENDMENTS

RELEASE 1.1 UPDATE, MARCH 1984

NEW DEVICE SUPPORT

- IBM 4248 Printer

  The FCB statement in IEBIMAGE can now be used to create forms control buffer modules in a form appropriate for use on the IBM 4248 Printer. Information to support the 4248 has been added to the IEBIMAGE chapter.

- IBM 3262 Model 5 Printer

  Information to support the IBM 3262 Model 5 Printer has been added to the IEBIMAGE chapter.

RELEASE 1.1, OCTOBER 1983

NEW DEVICE SUPPORT

- IBM 4245 Printer

  Information to support the IBM 4245 Printer has been added to the IEBIMAGE chapter.

- IBM 3800 Printing Subsystem Model 3

  The IBM 3800 Printing Subsystem Model 3 is supported by IEBIMAGE in full function mode.

NEW PROGRAMMING SUPPORT

- IEBCOPY Enhancements

  IEBCOPY can now be used to alter and copy load modules. Information to support the altering and copying functions, including the new ALTERMOD and COPYMOD statements, has been added to the IEBCOPY chapter.

- IEHMOVE Enhancements

  When used to move or copy sequential data sets on DASD, IEHMOVE now uses multiple BSAM buffers to improve performance. Information to support multiple buffers has been added to the IEHMOVE chapter.

CONTENTS

FIGURES

INTRODUCTION

MVS/370 Data Facility Product provides utility programs to assist in organizing and maintaining data. Each utility program falls into one of three classes of programs, determined by the function performed and the type of control of the utility.

SYSTEM UTILITY PROGRAMS

System utility programs are used to maintain and manipulate system and user data sets. Entire volume manipulation, for example, copying or restoring, is also provided. These programs must reside in an authorized library and are controlled by JCL statements and utility control statements.

They can be executed as jobs or can be invoked as subroutines by authorized programs. The invocation of utility programs and the linkage conventions are discussed in "Invoking Utility Programs from a Problem Program" on page 13.

Figure 1 is a list of system utility programs and their purpose.

System Utility	Purpose
IEHATLAS	To assign alternate tracks and recover usable data records when defective tracks are indicated
IEHINITT	To write standard labels on tape volumes
IEHLIST	To list system control data
IEHMOVE	To move or copy collections of data
IEHPROGM	To build and maintain system control data
IFHSTATR	To select, format, and write information about tape errors from the IFASMFDP tape or the SYS1.MAN data set.

Figure 1. System Utility Programs

DATA SET UTILITY PROGRAMS

Data set utility programs are used to reorganize, change, or compare data at the data set and/or record level. These programs are controlled by JCL statements and utility control statements.

These utilities manipulate partitioned, sequential, or indexed sequential data sets provided as input to the programs. Data ranging from fields within a logical record to entire data sets can be manipulated.

Data set utility programs can be executed as jobs or can be invoked as subroutines by a calling program. The invocation of utility programs and the linkage conventions are discussed in "Invoking Utility Programs from a Problem Program" on page 13.

Utility programs that manipulate data sets and are included in this manual cannot be used with VSAM data sets. Information about VSAM data sets can be found in VSAM Users Guide.

Two utilities, IEHMOVE and IEBCOPY, do not support Virtual Input/Output (VIO) data sets.

Figure 2 is a list of data set utility programs and their purpose.

Data Set Utility	Purpose
IEBCOMPR	To compare records in sequential or partitioned data sets
IEBCOPY	To copy, compress, or merge partitioned data sets, to add RLD count information to load modules, to select or exclude specified members in a copy operation, and to rename and/or replace selected members of partitioned data sets
IEBDG	To create a test data set consisting of patterned data
IEBEDIT	To selectively copy job steps and their associated JOB statements
IEBGENER	To copy records from a sequential data set or to convert a data set from sequential organization to partitioned organization
IEBIMAGE	To modify, print, or link modules for use with the IBM 3800 Printing Subsystem, the 3262 Model 5, or the 4248 printer
IEBISAM	To place source data from an indexed sequential data set into a sequential data set in a format suitable for subsequent reconstruction
IEBPTPCH	To print or punch records that reside in a sequential or partitioned data set
IEBTCRIN	To construct records from the input data stream that have been read from the IBM 2495 Tape Cartridge Reader
IEBUPDTE	To incorporate changes to sequential or partitioned data sets

Figure 2. Data Set Utility Programs

INDEPENDENT UTILITY PROGRAMS

Independent utility programs are used to prepare devices for system use when the operating system is not available. They operate outside of, and in support of, the operating system, are controlled by utility control statements, and cannot be invoked by a calling program. This publication addresses only the ICAPRTBL utility program.

The following figure shows the independent utility program and its purpose.

Independent Utility	Purpose
ICAPRTBL	To load the forms control and universal character set buffers of the IBM 3203-5 or 3211 printer after an unsuccessful attempt to IPL, with the 3203-5 or 3211 assigned as the output portion of a composite console.

Figure 3. Independent Utility Program

The selection of a specific program depends on the nature of the job to be performed. For example, renaming a data set involves modifying system control data. Therefore, a system utility program can be used to rename the data set. In some cases, a specific function can be performed by more than one program. "Guide to Utility Program Functions" on page 8 will help you find the program that performs the function you need.

DASD AND TAPE DEVICE SUPPORT

Except where noted, all the following DASD and tape devices are supported by all utility programs. Restrictions and peculiar device support are noted in the individual utility sections.

The table below indicates specific devices supported, and the notation to be used to reference them. The term **DASD** includes all direct access storage devices listed below.

	Device Number	Devices
DASD:	2305-1	2305-1
	2305-2	2305-2
	2314	2314
	2319	2319
	3330	3330-1, 3330-2, 3333, and 3350 in 3330-1 compatibility mode
	3330-1	3330-11, 3333-11, and 3350 in 3330-11 compatibility mode
	3330V	3850 MSS Virtual Volumes
	3340	3340, 3344 (both 35 & 70 megabyte models)
	3350	3350 Native mode
	3375	3375
	3380	3380
Tape:	2400	2400 (all models)
	2495	2495 (IEBTCRIN only)
	3400	3420 (all models)

CONTROL

System and data set utility programs are controlled by job control statements and utility control statements. The independent utility program is controlled by utility control statements only; because this program is independent of the operating system, job control statements are not required. The job control statements and utility control statements necessary to use utility programs are provided in the major discussion of each utility program.

JOB CONTROL STATEMENTS

A system or data set utility program can be introduced to the operating system in different ways:

- Job control statements can be included in the input stream.
- Job control statements, placed in a procedure library or defined as an inline procedure, can be included by means of the EXEC job control statement.
- A utility program can be invoked by a calling program.

If job control statements are placed in a procedure library, they should satisfy the requirements for most applications of the program; a procedure, of course, can be modified or supplemented for applications that require additional parameters, data sets, or devices. The data set utility IEBUPDTE can be used to enter a procedure into a procedure library; see "IEBUPDTE Program" on page 293.

A job that modifies a system data set (identified by SYS1.) must be run in a single job environment; however, a job that uses a system data set, but does not modify it, can be run in a multiprogramming environment. The operator should be informed of all jobs that modify system data sets.

DD statements should ensure that the volumes on which the data sets reside cannot be shared when update activity is being performed.

Job control statements can be continued on subsequent lines, but the continued line must begin in column 4 through 16. No continuation mark is required in column 72.

UTILITY CONTROL STATEMENTS

Utility control statements are used to identify a particular function to be performed by a utility program and, when required, to identify specific volumes or data sets to be processed.

The control statements for the utility programs have the following standard format:

label operation operand

The label symbolically identifies the control statement and, with the exception of system utility program IEHINITT, can be omitted. When included, a name must begin in the first position of the statement and must be followed by one or more blanks. It can contain from one to eight alphameric characters, the first of which must be alphabetic.

The operation identifies the type of control statement. It must be preceded and followed by one or more blanks.

The operand is made up of one or more keyword parameters separated by commas. The operand field must be preceded and followed by one or more blanks. Commas, parentheses, and blanks can be used only as delimiting characters.

Comments can be written in a utility statement, but they must be separated from the last parameter of the operand field by one or more blanks.

Continuing Utility Control Statements

Utility control statements are coded on cards or as online input and are contained in columns 1 through 71. A statement that exceeds 71 characters must be continued on one or more additional lines. A nonblank character must be placed in column 72 to indicate continuation. A utility statement can be interrupted either in column 71 or after any comma.

The continued portion of the utility control statement must begin in column 16 of the following statement.

Note: The IEHPROGM, IEBCOPY, IEBPTPCH, IEBGENER, IEBCOMPR, and IEBDG utility programs permit certain exceptions to these requirements (see the applicable program description).

The utility control statements are discussed in detail, as applicable, in the remaining chapters.

Restrictions

- Unless otherwise indicated in the description of a specific utility program, a temporary data set can be processed by a utility program only if the user specifies the complete name generated for the data set by the system (for example, DSNAME=SYS82296.T000051.RP001.JOBTEMP.TEMPMOD).
- The utility programs described in this book do not normally support VSAM data sets. For certain exceptions, refer to the various program descriptions.
- Most utility programs do not support ISCII/ASCII tape data sets. (Conversion from EBCDIC codes to ISCII/ASCII codes will result in loss of data.) Refer to the IEHINITT program for specific exceptions.

NOTATIONAL CONVENTIONS

A uniform system of notation describes the format of utility commands. This notation is not part of the language; it simply provides a basis for describing the structure of the commands.

The command format illustrations in this book use the following conventions:

- Brackets **[ ]** indicate an optional parameter.
- Braces **{ }** indicate a choice of entry; unless a default is indicated, you must choose one of the entries.

  Items separated by a vertical bar (**|**) represent alternative items. No more than one of these items may be selected.
- An ellipsis (...) indicates that multiple entries of the type immediately preceding the ellipsis are allowed.
- Other punctuation (parentheses, commas, spaces, etc.) must be entered as shown. A space is indicated by a blank.
- **BOLDFACE** type indicates the exact characters to be entered, except as described in the bulleted notes above. Such items must be entered exactly as illustrated.
- Lowercase underscored type specifies fields to be supplied by the user.

BOLDFACE UNDERSCORED type indicates a default option. If the parameter is omitted, the underscored value is assumed.

keyword=device=list

The term keyword is replaced by VOL, FROM, or TO.

The term device is replaced by either a generic name, for example, 3330; or an esoteric name, for example, DISK, if this esoteric name has been generated into your system. For DASD, the term list is replaced by one or more volume serial numbers separated by commas. When there is more than one volume serial number, the entire list field must be enclosed in parentheses.

For tapes, the term list is replaced by either one or more volume serial number/comma/data set sequence number pairs. Each pair is separated from the next pair by a comma. When there is more than one pair, the entire list field must be enclosed in parentheses; for example: FROM=3400=(tapeA,1,tapeB,1).

INSTALLATION CONSIDERATIONS

The System/370 versions of Device Support Facilities (Release 1 through 5) are not applicable for Data Facility Product Installations. The user must order and install Device Support Facilities Release 6 (5752-VS2) to run in an MVS/370 Data Facility Product environment.

Releases 1.0 and 1.1 of Data Facility Data Set Services (DFDSS) are not applicable for Data Facility Product installations. The user must install DFDSS Release 1.2 to run in an MVS/370 Data Facility Product environment. Installation of Release 1.2 supersedes Release 1.1.

The following utilities are not included as support for the Data Facility Product for MVS/370.

- IBCDASDI—Disk initialization functions are described in Device Support Facilities User's Guide and Reference.
- IBCDMPRS—Stand-alone disk restore functions are described in Data Facility Data Set Services User's Guide and Reference.
- IEHDASDR—Disk initialization functions are described in Device Support Facilities User's Guide and Reference. Dump restore functions are described in Data Facility Data Set Services User's Guide and Reference.

Note: DFDSS does not support the dump format produced by IEHDASDR or DRWDASDR.

- Analysis Program-1 (AP-1)—Functions to aid in the analysis of DASD errors are described in Device Support Facilities User's Guide and Reference.

SPECIAL REFERENCING AIDS

Two special referencing aids are included in this publication to help you locate the correct utility program for your needs and locate the correct example of the program for reference.

To locate the correct utility program, refer to Figure 5 on page 8 in "Guide to Utility Program Functions" on page 8.

To locate the right example, use the figure—called an "example directory"—that precedes each program's examples. Figure 4 on page 7 shows a portion of the example directory for IEHMOVE. The figure shows that IEHMOVE Example 1 is an example of moving a sequential data set and that IEHMOVE Example 2 is an example of copying a sequential data set.

Operation	Device	Comments	Example
MOVE Sequential	Disk	Source volume is demounted after job completion.	1
COPY Sequential	Disk	Three cataloged sequential data sets are to be copied. The disks are mountable.	2

Figure 4. Locating the Correct Example

Figure 5 shows a list of tasks that the utility programs can be used to perform. The left-hand column shows tasks that you might want to perform. The middle column more specifically defines the tasks. The right-hand column shows the utility programs that can be used for each task. Notice that in some cases more than one program may be available to perform the same task.

Task	Options	Utility Program
Add	a password	IEHPROGM
Alter in place	a load module	IEBCOPY
Assign alternate	tracks to a DASD volume and recover usable data	IEHATLAS
Catalog	a data set in an OS CVOL	IEHPROGM
Change	data set organization logical record length	IEBUPDTE IEBGENER
Compare	partitioned data sets sequential data sets records	IEBCOMPR
Compress in place	a partitioned data set	IEBCOPY
Construct	records from MTST and MTDI input	IEBTCRIN
Convert to partitioned	a sequential data set created as a result of an unload	IEBCOPY
	sequential data sets	IEBUPDTE, IEBGENER
Convert to sequential	a partitioned data set	IEBUPDTE, IEBCOPY
	an indexed sequential data set	IEBISAM, IEBDG
Copy	a direct access volume	IEHMOVE
	a load module	IEBCOPY
	a partitioned data set	IEBCOPY, IEHMOVE
	a volume of data sets	IEHMOVE
	an indexed sequential data set	IEBISAM
	job steps	IEBEDIT
	selected members	IEBCOPY, IEHMOVE

Figure 5 (Part 1 of 5). Tasks and Utility Programs

Task	Options	Utility Program
	sequential data sets	IEBGENER, IEHMOVE, IEBUPDTE
Create	a backup copy of a partitioned data set	IEBCOPY
	a character arrangement table module	IEBIMAGE
	a copy modification module	IEBIMAGE
	a 3800 or 4248 forms control buffer module	IEBIMAGE
	a graphic character modification module	IEBIMAGE
	a library character set module	IEBIMAGE
	a library of partitioned members	IEBUPDTE
	a member	IEBDG IEBGENER IEBUPDTE
	a sequential output data set	IEBDG
	an indexed sequential data set	IEBDG
	an output job stream	IEBEDIT
Delete	a password	IEHPROGM
	catalog entries	IEHPROGM
	records in a partitioned data set	IEBUPDTE
Edit	MTDI input	IEBTCRIN
Edit and convert to partitioned	a sequential data set	IEBGENER, IEBUPDTE
Edit and copy	a job stream	IEBEDIT
	a sequential data set	IEBGENER, IEBUPDTE
Edit and list	error statistics by volume (ESV) records	IFHSTATR
Edit and print	a sequential data set	IEBPTPCH
Edit and punch	a sequential data set	IEBPTPCH
Enter	a procedure into a procedure library	IEBUPDTE

Figure 5 (Part 2 of 5). Tasks and Utility Programs

Task	Options	Utility Program
Exclude	a partitioned data set member from a copy operation	IEBCOPY, IEHMOVE
Expand	a partitioned data set	IEBCOPY
	a sequential data set	IEBGENER
Generate	test data	IEBDG
Get	alternate tracks on a DASD volume	IEHATLAS
Include	changes to members or sequential data sets	IEBUPDTE
Insert records	into a partitioned data set	IEBUPDTE
Label	magnetic tape volumes	IEHINITT
List	a password entry	IEHPROGM
	a volume table of contents	IEHLIST
	number of unused directory blocks and tracks	IEBCOPY
	partitioned directories	IEHLIST
Load	a previously unloaded partitioned data set	IEBCOPY
	an indexed sequential data set	IEBISAM
	an unloaded data set	IEHMOVE
	UCS and FCB buffers of a 3211	ICAPRTBL
Merge	partitioned data sets	IEHMOVE, IEBCOPY
Modify	a partitioned or sequential data set	IEBUPDTE
Move	a volume of data sets	IEHMOVE
	partitioned data sets	IEHMOVE
	sequential data sets	IEHMOVE
Number records	in a new member	IEBUPDTE
	in a partitioned data set	IEBUPDTE
Password protect	add a password	IEHPROGM
	delete a password	IEHPROGM
	list passwords	IEHPROGM
	replace a password	IEHPROGM

Figure 5 (Part 3 of 5). Tasks and Utility Programs

Task	Options	Utility Program
Print	sequential data sets	IEBGENER, IEBUPDTE, IEBPTPCH
	partitioned data sets	IEBPTPCH
	selected records	IEBPTPCH
Punch	a partitioned data set member	IEBPTPCH
	a sequential data set	IEBPTPCH
	selected records	IEBPTPCH
Read	Tape Cartridge Reader input	IEBTCRIN
Reblock	a load module	IEBCOPY
	a partitioned data set	IEBCOPY
	a sequential data set	IEBGENER, IEBUPDTE
Recover	data from defective tracks on direct access volumes	IEHATLAS
Re-create	a partitioned data set	IEBCOPY
Rename	a partitioned data set member	IEBCOPY, IEHPROGM
	a sequential or partitioned data set	IEHPROGM
	moved or copied members	IEHMOVE
Renumber	logical records	IEBUPDTE
Replace	a password	IEHPROGM
	data on an alternate track	IEHATLAS
	identically named members	IEBCOPY
	logical records	IEBUPDTE
	members	IEBUPDTE
	records in a member	IEBUPDTE
	records in a partitioned data set	IEBUPDTE, IEBCOPY
	selected members	IEBCOPY
	selected members in a move or copy operation	IEBCOPY, IEHMOVE
Scratch	a volume table of contents	IEHPROGM
	data sets	IEHPROGM
Uncatalog	data sets	IEHPROGM

Figure 5 (Part 4 of 5). Tasks and Utility Programs

Task	Options	Utility Program
Unload	a partitioned data set	IEHMOVE, IEBCOPY
	a sequential data set	IEHMOVE
	an indexed sequential data set	IEBISAM
Update in place	a partitioned data set	IEBUPDTE

Figure 5 (Part 5 of 5). Tasks and Utility Programs

Utility programs can be invoked by a problem program through the use of the ATTACH or LINK macro instruction. In addition, IEBTCRIN can be invoked with the LOAD or CALL macro instruction.

The problem program must supply the following to the utility program:

- The information usually specified in the PARM parameter of the EXEC statement.
- The ddnames of the data sets to be used during processing by the utility program.

The following programs may execute authorized functions:

IEBCOPY, IEHATLAS, IEHINITT, IEHMOVE, IEHPROGM

When executing an authorized function, the calling program must be authorized via the Authorized Program Facility (APF).

When IEHMOVE, IEHPROGM, or IEHLIST is dynamically invoked in a job step containing a program other than one of these three, the DD statements defining mountable devices for the IEHMOVE, IEHPROGM, or IEHLIST program must be included in the job stream prior to DD statements defining data sets required by the other program.

LINK OR ATTACH MACRO INSTRUCTION

The LINK or ATTACH macro instruction can be used to invoke a utility program from a problem program.

The format of the LINK or ATTACH macro instruction is:

[label]	{LINK|ATTACH}	EP=progname ,PARAM=(optionaddr[,ddnameaddr] [,hdingaddr]) ,VL=1

where:

EP=progname
specifies the name of the utility program.

PARAM=
specifies, as a sublist, address parameters to be passed from the problem program to the utility program. These values can be coded:

optionaddr
specifies the address of an option list, OPTLIST, which is usually specified in the PARM parameter of the EXEC statement. This address must be written for all utility programs.

ddnameaddr
specifies the address of a list, DDNMELST, of alternate ddnames for the data sets used during utility program processing. If standard ddnames are used and this is not the last parameter in the list,

it should point to a halfword of zeros. If it is the last parameter, it may be omitted.

hdingaddr
specifies the address of a 6-byte list, HDNGLIST, which contains an EBCDIC page count for the output device. If hdingaddr is omitted, the page number defaults to 1.

VL=1
specifies that the sign bit of the last fullword of the address parameter list is to be set to 1.

Figure 6 shows these lists as they exist in the user's DC area. Note that the symbolic starting addresses for OPTLIST and DDNMELST fall on halfword boundaries which are not also fullword boundaries.

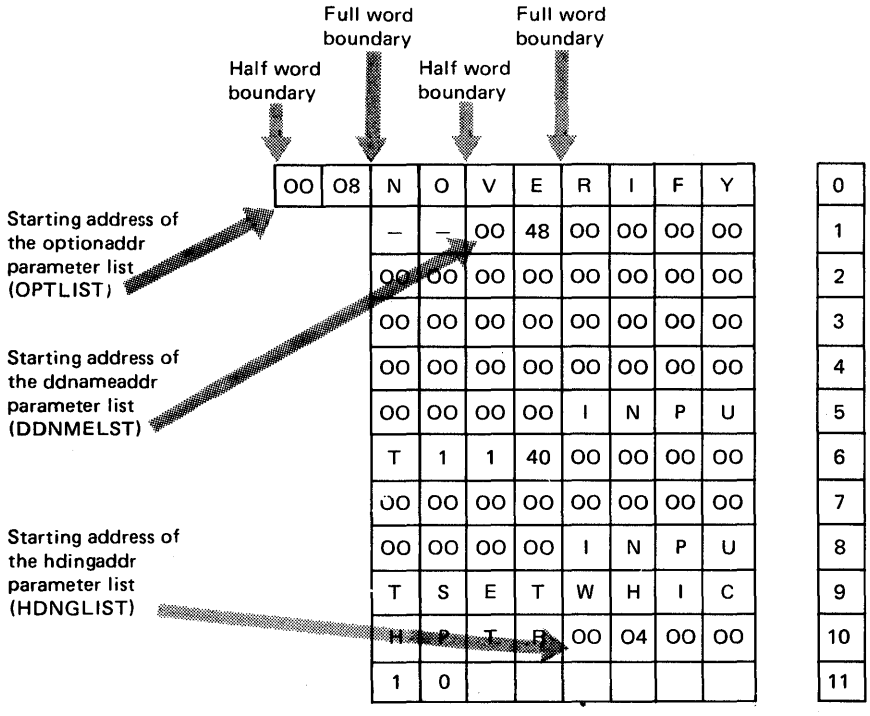


Figure 6. Typical Parameter Lists

The PARAM parameter of the LINK macro instruction in the calling program provides the utility program with the symbolic addresses of the parameter lists shown in Figure 6, as follows:

- The option list, OPTLIST, which includes the number of bytes in the list (hexadecimal 08) and the NOVERIFY option.
- The alternate ddname list, DDNMELST, which includes the number of bytes in the list (hexadecimal 48) and alternative names for the SYSIN INPUT11, SYSUT1 INPUTSET, and SYSUT2 WHICHPTR data sets.
- The heading list, HDNGLIST, which includes the number of bytes in the list (hexadecimal 04) and indicates the starting page number (shown as decimal 10) for printing operations controlled through the SYSPRINT data set.

The option list, OPTLIST, must begin on a halfword boundary that is not also a fullword boundary. The two high order bytes contain a hexadecimal count of the number of bytes in the remainder of the OPTLIST. (For all programs except IEHMOVE, IEHLIST, IEHPROGM, IEHINITT, IEBUPDTE, and IEBISAM, the count must be zero.) OPTLIST is free form with fields separated by commas. No blanks or zeros should appear in the list.

The ddname list, DDNMELST, must begin on a halfword boundary that is not also a fullword boundary. The two high order bytes contain a count of the number of bytes in the remainder of the list. Each name of fewer than 8 bytes must be left aligned and padded with blanks. If an alternate ddname is omitted from the list, the standard name is assumed. If the name is omitted within the list, the 8-byte entry must contain binary zeros. Names can be omitted from the end by merely shortening the list. Figure 7 shows the sequence of the 8-byte entries in the ddname list pointed to by ddnameaddr.

Entry	Standard Name
1	00000000
2	00000000
3	00000000
4	00000000
5	SYSIN
6	SYSPRINT
7	00000000
8	SYSUT1
9	SYSUT2
10	SYSUT3
11	SYSUT4

Figure 7. Sequence of DDNMELST Entries

The first 2 bytes of HDNGLIST contain the length in bytes of the heading list. The remaining 4 bytes contain a page number that the utility program is to place on the first page of printed output.

LOAD MACRO INSTRUCTION

IEBTCRIN can be invoked through use of the LOAD macro instruction.

The LOAD macro instruction causes the control program to bring the load module containing the specified entry point into main storage unless a copy is already there. Control is not passed to the load module.

The format of the LOAD macro instruction is:

[label]	**LOAD**	**{EP=IEBTCRIN|EPLOC=**address of name**}**

where:

EP=IEBTCRIN
is the entry point name of the program to be brought into main storage.

EPLOC=address of name
is the main storage address of the entry point name described above.

CALL MACRO INSTRUCTION

The CALL macro instruction can be used to pass control to IEBTCRIN after IEBTCRIN has been loaded into main storage.

Control can be passed to IEBTCRIN via a CALL macro instruction or via a branch and link instruction. If the branch and link instruction is used, register 1 must be loaded with the address of a parameter list of fullwords as described under "LINK or ATTACH Macro Instruction" on page 13. The last parameter list address must contain X'80' in byte 1 to indicate the last parameter in the list.

The format of the CALL macro instruction is:

[label]	CALL	IEBTCRIN[,optionaddr[,ddnameaddr] [,hdingaddr]) ,VL=1

where:

IEBTCRIN
is the name of the program to be given control; the name is used in the macro instruction as the operand of a V-type address constant.

optionaddr
specifies the address of an option list, OPTLIST, usually specified in the PARM parameter of the EXEC statement.

ddnameaddr
specifies the address of a list of alternate ddnames, DDNMELST, for the data sets used during IEBTCRIN processing. If standard ddnames are used and this is not the last parameter in the list, it should point to a halfword of zeros. If it is the last parameter, it may be omitted.

hdingaddr
specifies the address of a six-byte list, HDNGLIST, containing an EBCDIC page count for the output device.

VL=1
specifies that the high order bit of the last address parameter in the macro expansion is to be set to 1.

The option list, OPTLIST, must begin on a halfword boundary that is not also a fullword boundary. The two high order bytes contain a hexadecimal count of the number of bytes in the remainder of the OPTLIST. This count must be zero. OPTLIST is free form with fields separated by commas. No blanks or zeros should appear in the list.

The ddname list, DDNMELST, must begin on a halfword boundary that is not also a fullword boundary. The two high order bytes contain a count of the number of bytes in the remainder of the list. Each name of fewer than 8 bytes must be left aligned and

padded with blanks. If an alternate ddname is omitted from the list, the standard name is assumed. If the name is omitted within the list, the 8-byte entry must contain binary zeros. Names can be omitted from the end by merely shortening the list. Figure 7 on page 15 shows the sequence of the 8-byte entries in the ddname list pointed to by ddnameaddr.

The first two bytes of the heading list, HDNGLIST, contain the length in bytes of the heading list. The remaining four bytes contain a page number that IEBTCRIN places on the first page of printed output.

ICAPRTBL PROGRAM

ICAPRTBL is an independent utility that operates only in a System/370 environment. It is used to load the universal character set (UCS) buffer and the forms control buffer (FCB) for an IBM 3211 or 3203-5 Printer.

ICAPRTBL is used when the 3211/3203-5 is assigned as the output portion of a composite console and an unsuccessful attempt has been made to initialize the operating system because the UCS and FCB buffers contain improper bit patterns. ICAPRTBL properly loads the buffers so the operating system can be initialized.

Note: When an operable console printer keyboard is available, the buffers are loaded under the control of the operating system.

EXECUTING ICAPRTBL

ICAPRTBL must be loaded from a card reader. Control statements must follow the last card of the program. Only one printer can be initialized each time the program is executed.

To execute ICAPRTBL:

1. Mount the correct train on the printer and ready the printer.
2. Place the object program deck and the control cards in the card reader. Ready the reader and press the reader's END OF FILE key.
3. Load the object program from the reader by setting the load selector switches and pressing the console LOAD key.

Wait state codes will be displayed in the address portion of the PSW for normal termination and for input/output, system, or control card errors. Code B01 is issued for normal termination; B02 through B07 are issued for control card errors; B0A through B0C are issued for system errors; and B11 through B1D are issued for input/output errors. Figure 8 on page 19 shows these codes and their meanings.

Code	Meaning
B01	Visually check the train image printed on the 3211/3203-5.
B02	Missing control card or control card out of order.
B03	Incorrect JOB statement.
B04	Incorrect DFN statement.
B05	Incorrect UCS statement.
B06	Incorrect FCB statement.
B07	Incorrect END statement.
B0A	External interrupt.
B0B	Program check interrupt.
B0C	Machine check interrupt.
B11	Reader not online.
B12	Reader not ready.
B13	Reader unit check (display low virtual storage locations 2 through 7 for sense information).
B14	Reader channel error.
B15	No device end on reader.
B19	Printer not online.
B1A	Printer not ready.
B1B	Printer unit check (display low virtual storage locations 2 through 7 for sense information).
B1C	Printer channel error.
B1D	No device end on printer.

Figure 8. ICAPRTBL Wait-State Codes

INPUT AND OUTPUT

ICAPRTBL uses, as input, utility control statements that contain images to be loaded into the universal character set and/or the forms control buffer. ICAPRTBL produces, as output, properly loaded UCS and FCB buffers.

CONTROL

ICAPRTBL is controlled by utility control statements. Because ICAPRTBL is an independent utility program, operating system job control statements are not used.

UTILITY CONTROL STATEMENTS

All utility control statement operands must be preceded and followed by one or more blanks. Continuation requirements for utility control statements are described in "Continuing Utility Control Statements" on page 5.

ICAPRTBL utility control statements are listed below.

Statement	Use
JOB	Indicates the beginning of an ICAPRTBL job.
DFN	Defines the address of the 3211 or 3203-5, specifies that lowercase letters are to be printed in uppercase when the lowercase print train is not available, and identifies UCS and FCB image names.
UCS	Contains an image of the characters to be loaded into the UCS buffer.
FCB	Defines the image to be loaded into the FCB.
END	Indicates the end of an ICAPRTBL job.

Figure 9. ICAPRTBL Utility Control Statements

JOB Statement

The JOB statement indicates the beginning of an ICAPRTBL job.

The format of the JOB statement is:

[label]	**JOB**	[user-information]

DFN Statement

The DFN statement is used to define the address of the 3211 or 3203-5, to specify that lowercase letters are to be printed in uppercase when the lowercase print train is not available, and to identify UCS and FCB image names.

The format of the DFN statement is:

DFN	**ADDR=**cuu
	[,FOLD=Y|N]
	[,DEVT=3211|3203-5]
	[,UCS=ucsname**|AN|A11]**
	[,FCB=fcbname**|STD|STD2]**

UCS Statement

The UCS statement contains an image to be loaded into the UCS buffer.

The format of the UCS statement is:

[ucsname]	UCS	ucs-image

FCB Statement

The FCB statement defines the image to be loaded into the forms control buffer. The FCB statement may precede or follow the UCS statement.

The format of the FCB statement is:

[fcbname]	FCB	LPI={6|8} ,LNCH={(l,c)[,(l,c)...]} ,FORMEND=x

END Statement

The END statement signals the end of the ICAPRTBL job.

The format of the END statement is:

[label]	END	[user-information]

Parameters	Applicable Control Statements	Description of Parameters
ADDR	DFN	**ADDR=cuu** specifies the channel number, c, and unit number, uu, of the 3211 or 3203-5.
DEVT	DFN	**DEVT=3211|3203-5** specifies the device type for which the ADDR parameter applies. 3211 is the default device type.
FCB	DFN	**FCB=fcbname|STD|STD2** specifies a 1 to 8 character name of the image loaded into the forms control buffer. The actual image loaded into the buffer is not affected by this name, but serves as a meaningful reference when printed on the printer. fcbname should be the same as the FCB image being used. **STD2** is the default.
FOLD	DFN	**FOLD=Y|N** specifies whether lowercase letters are to be printed as uppercase letters when the lowercase print train is not available. The values can be coded: **Y** specifies that lowercase letters are to be printed as uppercase letters when the lowercase print train is not available. **N** specifies that lowercase letters are not to be printed as uppercase letters. This is the default.
FORMEND	FCB	**FORMEND=x** specifies the number of lines (maximum 180) on the printer form. For an 11-inch form, spacing six lines per inch, x must be 66.
LNCH	FCB	**LNCH=((l,c)[,(l,c)...])** specifies the channels of the FCB image. Each set of parentheses must contain the line number (1-180), a comma, and the channel number (1-12) to be assigned to that line. One or all of the 12 channels may be assigned in any order. Each set must be separated by commas and the entire group surrounded by parentheses.

Parameters	Applicable Control Statements	Description of Parameters
LPI	FCB	**LPI={6|8}** specifies the number of lines per inch that will be printed on the document. These values can be coded: **6** specifies that six lines per inch will be printed. **8** specifies that eight lines per inch will be printed.
UCS	DFN	**UCS=**ucsname|**AN**|**A11** is a 1 to 8 character alphameric name of the image loaded into the UCS buffer. This name is printed on the printer to serve as a reference to the print train being used. **AN** is the default for 3203-5 devices. **A11** is the default for 3211 devices.
ucs-image	UCS	ucs-image specifies characters to be loaded into the UCS buffer. The characters must be contained in columns 16 through 71. The first UCS statement contains the first 56 characters; subsequent statements contain continuations of the image to be loaded into the UCS buffer. A continuation mark is required in column 72 of a continued UCS image card.
user-information	JOB END	[user-information] specifies user explanation of action and comments.

ICAPRTBL EXAMPLES

The examples that follow illustrate some of the uses of ICAPRTBL. Figure 10 can be used as a quick-reference guide to the examples. The numbers in the "Examples" column refer to examples that follow.

Devices	Examples
3211	1, 2
3203-5	3, 4

Figure 10. ICAPRTBL Example Directory

ICAPRTBL EXAMPLE 1

In this example, a 3211 UCS image (A11) and an FCB image are loaded into the UCS and FCB buffers.

```
                                                                    72
JOB LOAD A11 IMAGE
DFN ADDR=002,FOLD=N
A11 UCS 1<.=IHGFEDCBA*$-RQPONMLKJ%,&ZYXWVUTS/@#0987654321<.=IHGF
        EDCBA*$-RQPONMLKJ%,&ZYXWVUTS/@#0987654321<.=IHGFEDCBA*$-
        RQPONMLKJ%,&ZYXWVUTS/@#0987654321<.=IHGFEDCBA*$-RQPONMLK
        J%,&ZYXWVUTS/@#0987654321<.=IHGFEDCBA*$-RQPONMLKJ%,&ZYXW
        VUTS/@#0987654321<.=IHGFEDCBA*$-RQPONMLKJ%,&ZYXWVUTS/@#0
        987654321<.=IHGFEDCBA*$-RQPONMLKJ%,&ZYXWVUTS/23098765432
        1<.=IHGFEDCBA*$-RQPONMLKJ%,&ZYXWVUTS/@#0987654321<.=IHGF
        EDCBA*$-RQPONMLKJ%,&ZYXWVUTS/@#098765432
STD2 FCB  LPI=6,                                                    C
          LNCH=((4,1),(10,2),(16,3),(22,4),(28,5),(34,6),(40,7),    C
          (46,8),(52,10),(58,11),(64,12),(66,9)),                   C
          FORMEND=66
END
```

The control statements are discussed below:

- DFN specifies the channel and unit number of the default device type 3211 and FOLD=N specifies that lowercase letters are not to be printed as uppercase letters when the lowercase print train is not available.
- UCS specifies the characters to be loaded into the UCS buffer.
- FCB specifies the values to be loaded into the forms control buffer. LPI=6 indicates that six lines per inch will be printed, and FORMEND=66 specifies 66 lines per page.

ICAPRTBL EXAMPLE 2

In this example, a 3211 UCS image (P11) and an IBM standard FCB image are loaded into the UCS and FCB buffers by specifying images via the UCS and FCB parameters of the DFN statement.

```
JOB LOAD 3211 P11 IMAGE
DFN UCS=P11,ADDR=004,FCB=STD
END
```

The DFN control statement is discussed below:

- By omitting the DEVT parameter, the default device type is 3211.
- The UCS parameter specifies the UCS image ID to be loaded into the UCS buffer from standard image tables provided by the utility.
- The ADDR parameter specifies the channel and unit number of the 3211.
- By omitting the FOLD parameter, the default FOLD value N is selected, specifying that lowercase letters are not to be printed as uppercase letters when the lowercase print train is not available.

The FCB parameter specifies the standard FCB image id (STD) to be loaded into the FCB buffer from standard image tables provided by the utility.

ICAPRTBL EXAMPLE 3

In this example, a 3203-5 UCS image (AN by default) and a standard FCB image (STD2 by default) are loaded into the UCS and FCB buffers.

```
JOB
DFN DEVT=3203-5,ADDR=002
END
```

The DFN statement is discussed below:

- The DEVT parameter specifies the device type as 3203-5.
- The ADDR parameter specifies the channel and unit number of the 3203-5.
- By omitting the FOLD parameter, the default FOLD value N is selected specifying that lowercase letters are not to be printed as uppercase letters when the lowercase print train is not available.
- By omitting both a UCS statement and the UCS parameter, the default 3203-5 UCS image (AN) is loaded into the UCB buffer from standard image tables provided by the utility.
- By omitting both an FCB statement and the FCB parameter, the default FCB image (STD2) is loaded into the FCB buffer from standard image tables provided by the utility.

ICAPRTBL EXAMPLE 4

In this example, a 3203-5 UCS image (AN by default) and a provided FCB image are loaded, respectively, into the UCS and FCB buffers.

```
                                                                      72
      JOB 3203-5 USER FCB
USER  FCB FORMEND=88,LPI=8,LNCH=((4,1),(12,2),                        C
          (20,3),(28,4),(36,5),(44,6),(52,7),                         C
          (60,8),(68,10),(76,11),(84,12),(88,9))
      DFN FOLD=Y,                                                     C
          FCB=STD,                                                    C
          ADDR=003,                                                   C
          DEVT=3203-5
      END
```

The control statements are discussed below:

- The JOB statement includes user comments on the action taken.
- The FCB statement specifies the values to be loaded into the forms control buffer. FORMEND=88 and LPI=8 indicate that there will be 88 lines per page, 8 lines per inch. Note that the specification of the FCB parameter on the DFN statement is overridden by the FCB statement specification.

- The DEVT parameter of the DFN statement specifies the device type as 3203-5.
- The ADDR parameter specifies the channel and unit number of the 3203-5.
- The FOLD=Y parameter specifies that lowercase letters are to be printed as uppercase letters when the lowercase print train is not available.
- By omitting both a UCS statement and the UCS parameter of the DFN statement, the default 3203-5 UCS image (AN) is loaded from standard image tables provided by the utility.

IEBCOMPR is a data set utility used to compare two sequential or two partitioned data sets at the logical record level to verify a backup copy. Fixed, variable, or undefined records from blocked or unblocked data sets or members can also be compared.

Two sequential data sets are considered **equal**, that is, are considered to be identical, if:

- The data sets contain the same number of records, and,
- Corresponding records and keys are identical.

Two partitioned data sets are considered equal if:

- Corresponding members contain the same number of records.
- Note lists are in the same position within corresponding members.
- Corresponding records and keys are identical.

If all of these conditions are not met for a specific type of data set, an unequal comparison results. If records are unequal, the record and block numbers, the names of the DD statements that define the data sets, and the unequal records are listed in a message data set. Ten successive unequal comparisons terminate the job step unless a user routine is provided to handle error conditions.

Partitioned data sets can be compared only if all the names in one or both of the directories have counterpart entries in the other directory. The comparison is made on members identified by these entries and corresponding user data.

Figure 11 shows the directories of two partitioned data sets. Directory 2 contains corresponding entries for all the names in Directory 1; therefore, the data sets can be compared.

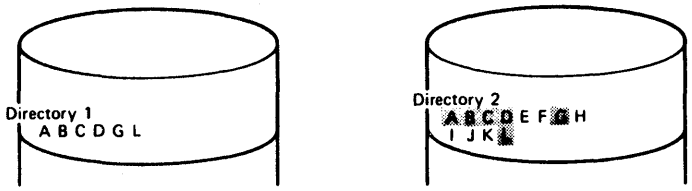


Figure 11. Partitioned Directories Whose Data Sets Can Be Compared Using IEBCOMPR

Figure 12 on page 28 shows the directories of two partitioned data sets. Each directory contains a name that has no corresponding entry in the other directory; therefore, the data sets cannot be compared, and the job step is terminated.

User exits are provided for optional user routines to process user labels, handle error conditions, and modify source records. See Appendix A, "Exit Routine Linkage" on page 438 for a discussion of the linkage conventions to be followed when user routines are used.

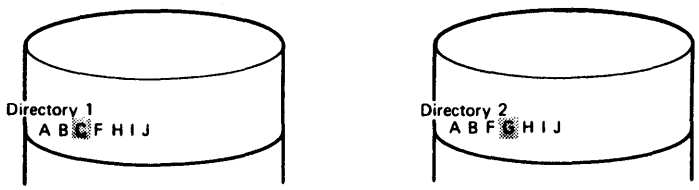


Figure 12. Partitioned Directories Whose Data Sets Cannot Be Compared Using IEBCOMPR

INPUT AND OUTPUT

IEBCOMPR uses the following input:

- Two sequential or two partitioned data sets to be compared.
- A control data set that contains utility control statements. This data set is required if the input data sets are partitioned or if user routines are used.

IEBCOMPR produces as output a message data set that contains informational messages (for example, the contents of utility control statements), the results of comparisons, and error messages.

RETURN CODES

IEBCOMPR returns a code in register 15 to indicate the results of program execution. The return codes and their meanings are listed below.

Codes	Meaning
00 (00 hex)	Successful completion.
08 (08)	An unequal comparison. Processing continues.
12 (0C)	An unrecoverable error exists. The job step is terminated.
16 (10)	A user routine passed a return code of 16 to IEBCOMPR. The job step is terminated.

Figure 13. IEBCOMPR Return Codes

CONTROL

IEBCOMPR is controlled by job control statements and utility control statements. The job control statements are required to execute or invoke IEBCOMPR and to define the data sets that are used and produced by IEBCOMPR. The utility control statements are used to indicate the input data set organization (that is, sequential or partitioned), to identify any user routines that may be provided, and to indicate whether user labels are to be treated as data.

JOB CONTROL STATEMENTS

Figure 14 shows the job control statements for IEBCOMPR.

One or both of the input data sets can be passed from a preceding job step.

Input data sets residing on different device types can be compared. Input data sets with a sequential organization written at different densities can also be compared.

Statement	Use
JOB	Initiates the job.
EXEC	Specifies the program name (PGM=IEBCOMPR) or, if the job control statements reside in a procedure library, the procedure name.
SYSPRINT DD	Defines a sequential message data set, which can be written to a system output device, a tape volume, or a direct access volume.
SYSUT1 DD	Defines an input data set to be compared.
SYSUT2 DD	Defines an input data set to be compared.
SYSIN DD	Defines the control data set or specifies DUMMY if the input data sets are sequential and no user routines are provided. The control data set normally resides in the input stream; however, it can be defined as a member within a library of partitioned members.

Figure 14. Job Control Statements for IEBCOMPR

The SYSPRINT DD statement must be present for each use of IEBCOMPR. The block size specified in the SYSPRINT DD statement must be a multiple of 121.

The SYSIN DD statement is required. The block size specified in the SYSIN DD statement must be a multiple of 80.

The logical record lengths of the input data sets must be identical; otherwise, unequal comparisons result. The block sizes of the input data sets can differ; however, block sizes must be multiples of the logical record length.

UTILITY CONTROL STATEMENTS

The utility control statements used to control IEBCOMPR are:

Statement	Use
COMPARE	Indicates the organization of a data set.
EXITS	Identifies user exit routines to be used.
LABELS	Indicates whether user labels are to be treated as data by IEBCOMPR.

Figure 15. IEBCOMPR Utility Control Statements

Continuation requirements for utility control statements are described in "Continuing Utility Control Statements" on page 5.

COMPARE Statement

The COMPARE statement is used to indicate the organization of data sets to be compared.

The COMPARE statement, if included, must be the first utility control statement. COMPARE is required if the EXITS or LABELS statement is used or if the input data sets are partitioned data sets.

The format of the COMPARE statement is:

[label]	COMPARE	TYPORG={PS|PO}

EXITS Statement

The EXITS statement is used to identify any user exit routines to be used. If a user exit routine is used, the EXITS statement is required. If more than one valid EXITS statement is included, all but the last EXITS statement are ignored. For a discussion of the processing of user labels as data set descriptors, see Appendix C, "Processing User Labels" on page 446.

The format of the EXITS statement is:

[label]	EXITS	[INHDR=routinename] [,INTLR=routinename] [,ERROR=routinename] [,PRECOMP=routinename]

LABELS Statement

The LABELS statement specifies whether user labels are to be treated as data by IEBCOMPR. For a discussion of this option, refer to Appendix C, "Processing User Labels" on page 446.

The format of the LABELS statement is:

[label]	LABELS	[DATA={YES|NO|ALL|ONLY}]

Note: LABELS DATA=NO must be specified to make IBM standard/user label (SUL) exits inactive when input/output data sets with nonstandard labels (NSL) are to be processed.

If more than one valid LABELS statement is included, all but the last LABELS statement are ignored.

Parameters	Applicable Control Statements	Description of Parameters
DATA	LABELS	**DATA={YES|NO|ALL|ONLY}** specifies whether user labels are to be treated as data. The values that can be coded are: **YES** specifies that any user labels that are not rejected by a user's label processing routine are to be treated as data. Processing of labels as data stops in compliance with standard return codes. YES is the default. **NO** specifies that user labels are not to be treated as data. **ALL** specifies that all user labels are to be treated as data. A return code of 16 causes IEBCOMPR to complete processing of the remainder of the group of user labels and to terminate the job step. **ONLY** specifies that only user header labels are to be treated as data. User header labels are processed as data regardless of any return code. The job terminates upon return from the OPEN routine.
ERROR	EXITS	**ERROR=routinename** specifies the name of the routine that is to receive control after each unequal comparison for error handling. If this parameter is omitted and ten consecutive unequal comparisons occur while IEBCOMPR is comparing sequential data sets, processing is terminated; if the input data sets are partitioned, processing continues with the next member.

Parameters	Applicable Control Statements	Description of Parameters
INHDR	EXITS	**INHDR=**routinename specifies the name of the routine that processes user input header labels.
INTLR	EXITS	**INTLR=**routinename specifies the name of the routine that processes user input trailer labels.
PRECOMP	EXITS	**PRECOMP=**routinename specifies the name of the routine that processes logical records (physical blocks in the case of variable spanned (VS) or variable blocked spanned (VBS) records longer than 32K bytes) from either or both of the input data sets before they are compared.
TYPORG	COMPARE	**TYPORG={**PS**|PO}** specifies the organization of the input data sets. The values that can be coded are: **PS** specifies that the input data sets are sequential data sets. This is the default. **PO** specifies that the input data sets are partitioned data sets.

IEBCOMPR EXAMPLES

The examples in Figure 16 illustrate some of the uses of IEBCOMPR. The numbers in the "Example" column refer to examples that follow.

Examples that use **disk** or **tape** in place of actual device numbers must be changed before use. See "DASD and Tape Device Support" on page 3 for valid device number notation.

Operation	Data Set Organization	Devices	Comments	Example
COMPARE	Sequential	9-track Tape	No user routines. Blocked input.	1
COMPARE	Sequential	7-track Tape	No user routines. Blocked input.	2
COMPARE	Sequential	7-track Tape and 9-track Tape	User routines. Blocked input. Different density tapes.	3

Figure 16 (Part 1 of 2). IEBCOMPR Example Directory

Operation	Data Set Organization	Devices	Comments	Example
COMPARE	Sequential	Card Reader, 9-track Tape	No user routines. Blocked input.	4
COMPARE	Partitioned	Disk	No user routines. Blocked input.	5
COPY (using IEBCOPY) and COMPARE	Sequential	9-track Tape	No user routines. Blocked input. Two job steps; data sets are passed to second job step.	6
COPY (using IEBCOPY) and COMPARE	Partitioned	Disk	User routine. Blocked input. Two job steps; data sets are passed to second job step.	7

Figure 16 (Part 2 of 2). IEBCOMPR Example Directory

IEBCOMPR EXAMPLE 1

In this example, two sequential data sets that reside on 9-track tape volumes are to be compared.

```
//TAPETAPE JOB  09#660,SMITH
//         EXEC PGM=IEBCOMPR
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   UNIT=tape,LABEL=(,NL),
//              DCB=(RECFM=FB,LRECL=80,BLKSIZE=2000),
//              DISP=(OLD,KEEP),VOLUME=SER=001234
//SYSUT2   DD   UNIT=tape,LABEL=(,NL),DISP=(OLD,KEEP),
//              DCB=(RECFM=FB,LRECL=80,BLKSIZE=1040),
//              VOLUME=SER=001235
//SYSIN    DD   DUMMY
/*
```

Because no user routines are used and the input data sets have a sequential organization, utility control statements are not necessary.

The job control statements are discussed below:

- SYSUT1 DD defines an input data set, which resides on an unlabeled, 9-track tape volume.
- SYSUT2 DD defines an input data set, which resides on an unlabeled, 9-track tape volume.
- SYSIN DD defines a dummy data set.

IEBCOMPR EXAMPLE 2

In this example, two sequential data sets that reside on 7-track tape volumes are compared.

```
//TAPETAPE JOB  09#660,SMITH
//         EXEC PGM=IEBCOMPR
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=SET1,LABEL=(2,SUL),DISP=(OLD,KEEP),
//              VOL=SER=001234,DCB=(DEN=2,RECFM=FB,LRECL=80,
//              BLKSIZE=2000,TRTCH=C),UNIT=3400
//SYSUT2   DD   DSNAME=SET2,LABEL=(,SUL),DISP=(OLD,KEEP),
//              VOL=SER=001235,DCB=(DEN=2,RECFM=FB,LRECL=80,
//              BLKSIZE=2000,TRTCH=C),UNIT=3400
//SYSIN    DD   *
      COMPARE   TYPORG=PS
       LABELS   DATA=ONLY
/*
```

The control statements are discussed below:

- SYSUT1 DD defines an input data set, SET1, which resides on a labeled, 7-track tape volume. The blocked data set was originally written at a density of 800 bits per inch (DEN=2) with the data converter on (TRTCH=C).
- SYSUT2 DD defines an input data set, SET2, which is the first or only data set on a labeled, 7-track tape volume. The blocked data set was originally written at a density of 800 bits per inch (DEN=2) with the data converter on (TRTCH=C).
- SYSIN DD defines the control data set, which follows in the input stream.
- COMPARE TYPORG=PS specifies that the input data sets are sequentially organized.
- LABELS DATA=ONLY specifies that user header labels are to be treated as data and compared. All other labels on the tape are ignored.

IEBCOMPR EXAMPLE 3

In this example, two sequential data sets written at different densities on different tape units are compared.

```
//TAPETAPE JOB  09#660,SMITH
//         EXEC PGM=IEBCOMPR
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=SET1,LABEL=(,SUL),DISP=(OLD,KEEP),
//              VOL=SER=001234,DCB=(DEN=1,RECFM=FB,LRECL=80,
//              BLKSIZE=320,TRTCH=C),UNIT=3400
//SYSUT2   DD   DSNAME=SET2,LABEL=(,SUL),DISP=(OLD,KEEP),
//              DCB=(RECFM=FB,LRECL=80,BLKSIZE=640),
//              UNIT=tape,VOLUME=SER=001235
//SYSIN    DD   *
      COMPARE   TYPORG=PS
        EXITS   INHDR=HDRS,INTLR=TLRS
       LABELS   DATA=NO
/*
```

The control statements are discussed below:

- SYSUT1 DD defines an input data set, SET1, which is the first or only data set on a labeled, 7-track tape volume. The blocked data set was originally written at a density of 556 bits per inch (DEN=2) with the data converter on (TRTCH=C).
- SYSUT2 DD defines an input data set, SET2, which is the first or only blocked data set on a labeled tape volume. In this example, assume SYSUT2 is on a 9-track tape drive.
- SYSIN DD defines the control data set, which follows in the input stream.
- COMPARE TYPORG=PS specifies that the input data sets are sequentially organized.
- EXITS identifies the names of routines to be used to process user input header labels and trailer labels.
- LABELS DATA=NO specifies that the user input header and trailer labels for each data set are not to be compared.

IEBCOMPR EXAMPLE 4

In this example, two sequential data sets (card input and tape input) are compared.

```
//CARDTAPE JOB  09#660,SMITH
//        EXEC PGM=IEBCOMPR
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   DUMMY
//SYSUT2   DD   UNIT=tape,VOLUME=SER=001234,LABEL=(,NL),
//              DCB=(RECFM=FB,LRECL=80,BLKSIZE=2000),
//              DISP=(OLD,KEEP)
//SYSUT1   DD   DATA

(input card data set)

/*
```

The control statements are discussed below:

- SYSIN DD defines a dummy control data set. Because no user routines are provided and the input data sets are sequential, utility control statements are not necessary.
- SYSUT2 DD defines an input data set, which resides on an unlabeled, 9-track tape volume.
- SYSUT1 DD defines an input data set (card input).

IEBCOMPR EXAMPLE 5

In this example, two partitioned data sets are compared.

```
//DISKDISK JOB  09#660,SMITH
//         EXEC PGM=IEBCOMPR
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=PDSSET1,UNIT=disk,DISP=SHR,
//              DCB=(RECFM=FB,LRECL=80,BLKSIZE=2000),
//              VOLUME=SER=111112
//SYSUT2   DD   DSNAME=PDSSET2,UNIT=disk,DISP=SHR,
//              DCB=(RECFM=FB,LRECL=80,BLKSIZE=2000),
//              VOLUME=SER=111113
//SYSIN    DD   *
     COMPARE   TYPORG=PO
/*
```

The control statements are discussed below:

- SYSUT1 DD defines an input partitioned data set, PDSSET1. The blocked data set resides on a disk volume.
- SYSUT2 DD defines an input partitioned data set, PDSSET2. The blocked data set resides on a disk volume.
- SYSIN DD defines the control data set, which follows in the input stream.
- COMPARE TYPORG=PO indicates that the input data sets are partitioned.

IEBCOMPR EXAMPLE 6

In this example, a sequential data set is copied and compared in two job steps.

```
//TAPETAPE JOB  09#660,SMITH
//STEPA    EXEC PGM=IEBCOPY
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSN=COPYSET1,UNIT=tape,
//              DISP=(OLD,PASS),
//              DCB=(RECFM=FB,LRECL=80,BLKSIZE=640),
//              LABEL=(,SL),
//              VOLUME=SER=001234
//SYSUT2   DD   DSNAME=COPYSET2,DISP=(,PASS),LABEL=(,SL),
//              DCB=(RECFM=FB,LRECL=80,BLKSIZE=640),
//              UNIT=tape,
//              VOLUME=SER=001235
//SYSIN    DD   DUMMY
/*
//STEPB    EXEC PGM=IEBCOMPR
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=*.STEPA.SYSUT1,DISP=(OLD,KEEP)
//SYSUT2   DD   DSNAME=*.STEPA.SYSUT2,DISP=(OLD,KEEP)
//SYSIN    DD   DUMMY
/*
```

The first job step copies the data set and passes the original and copied data sets to the second job step. The second job step compares the two data sets.

The control statements for the IEBCOMPR job step are discussed below:

- SYSUT1 DD defines an input data set passed from the preceding job step (COPYSET1). The data set resides on a labeled, 9-track tape volume.
- SYSUT2 DD defines an input data set passed from the preceding job step. (COPYSET2). The data set, which was created in the preceding job step, resides on a labeled, 9-track tape volume.
- SYSIN DD defines a dummy control data set. Because the input is sequential and no user exits are provided, no utility control statements are required.

IEBCOMPR EXAMPLE 7

In this example, a partitioned data set is copied and compared in two job steps.

The example follows:

```
//DISKDISK JOB  09#660,SMITH
//STEPA    EXEC PGM=IEBCOPY
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=OLDSET,UNIT=disk,DISP=SHR,
//              VOLUME=SER=111112,
//              DCB=(RECFM=FB,LRECL=80,BLKSIZE=640)
//SYSUT2   DD   DSNAME=NEWMEMS,UNIT=disk,DISP=(,PASS),
//              VOLUME=SER=111113,SPACE=(TRK,(5,5,5)),
//              DCB=(RECFM=FB,LRECL=80,BLKSIZE=640)
//SYSUT3   DD   UNIT=SYSDA,SPACE=(TRK,(1))
//SYSUT4   DD   UNIT=SYSDA,SPACE=(TRK,(1))
//SYSIN    DD   *
         COPY OUTDD=SYSUT2,INDD=SYSUT1
       SELECT MEMBER=(A,B,D,E,F)
/*
//STEPB    EXEC PGM=IEBCOMPR
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=OLDSET,DISP=(OLD,KEEP)
//SYSUT2   DD   DSNAME=NEWMEMS,DISP=(OLD,KEEP)
//SYSIN    DD   *
      COMPARE TYPORG=PO
        EXITS ERROR=SEEERROR
/*
```

The first job step copies the data set and passes the original and copied data sets to the second job step. The second job step compares the two data sets.

The control statements for the IEBCOMPR job step are discussed below:

- SYSUT1 DD defines a blocked input data set (OLDSET) that is passed from the preceding job step. The data set resides on a disk volume.
- SYSUT2 DD defines a blocked input data set (NEWMEMS) that is passed from the preceding job step. The data set resides on a disk volume.

- SYSUT3 and SYSUT4 define temporary system data sets to be used for work files during IEBCOPY. These are not passed to IEBCOMPR.
- SYSIN DD defines the control data set, which follows in the input stream.
- COMPARE TYPORG=PO specifies partitioned organization.
- EXITS specifies that a user error routine, SEEERROR, is to be used.

Because the input data set names are not identical, the data sets can be retrieved by their data set names.

IEBCOPY is a data set utility used to copy one or more partitioned data sets or to merge partitioned data sets. A partitioned data set that is copied to a sequential data set is said to be **unloaded**. The sequential data set created by an unload operation can be copied to any direct access storage device. When one or more data sets created by an unload operation are used to re-create a partitioned data set, this is called a **load** operation. Specific members of a partitioned or unloaded data set can be selected for, or excluded from, a copy, unload, or load process.

IEBCOPY can be used to:

- Create a backup copy of a partitioned data set.
- Copy one or more data sets per copy operation.
- Copy one partitioned data set to a sequential data set (unload).
- Copy one or more data sets created by an unload operation to any direct access device (load).
- Select members from a data set to be copied, unloaded, or loaded.
- Replace identically named members on data sets (except when unloading).
- Replace selected data set members.
- Rename selected members.
- Exclude members from a data set to be copied, unloaded, or loaded.
- Compress partitioned data sets in place (except when the data set is an unloaded data set).
- Merge data sets (except when unloading).
- Re-create a data set that has exhausted its primary, secondary, or directory space allocation.
- Alter load modules in place.
- Copy and reblock load modules.

In addition, IEBCOPY automatically lists the number of unused directory blocks and the number of unused tracks available for member records in the output partitioned data set. If LIST=NO is coded (see "COPY Statement" on page 51), the names of copied, unloaded, or loaded members listed by the input data set are suppressed.

CREATING A BACKUP COPY

IEBCOPY can be used to create a backup copy of a partitioned data set by copying (unloading) it to a sequential data set. A partitioned data set can be totally or partially unloaded to any tape volume or direct access device supported by BSAM. A data set is unloaded when physical sequential organization space allocation is specified for the output data set on a direct access device or when the output data set is a tape volume. To unload more than one partitioned data set to the same volume in one execution of IEBCOPY, multiple copy operations must be used

and multiple sequential data sets must be allocated on the same volume.

A data set with a physical sequential organization resulting from an unload operation can, in turn, be copied. No output tape file will be created if the input is a null file.

COPYING DATA SETS

IEBCOPY can be used to copy a partitioned data set, totally or in part, from one direct access volume to another. In addition, a data set can be copied to its own volume, provided its data set name is changed. If the data set name is not changed, the data set is compressed in place.

Note that copied members are not reordered. Members are copied in the order in which they exist on the original data set. If the members are to be reordered, IEHMOVE can be used for the copy operation (see "IEHMOVE Program" on page 361).

COPYING OR LOADING UNLOADED DATA SETS

Data sets can be copied or loaded, totally or in part, from one or more direct access volumes or tape volumes to a single direct access volume. To copy or load more than one input partitioned data set, specify more than one input data set with the COPY statement. The input data sets are copied or loaded in the order in which they are specified.

SELECTING MEMBERS TO BE COPIED, UNLOADED, OR LOADED

Members can be selected from one or more input data sets. Selected members can be copied, unloaded, or loaded from the input data sets specified on the INDD statement preceding a SELECT statement.

Selected members are searched for in a low-to-high (a-to-z) collating sequence, regardless of the order in which they are specified; however, they are copied in the same physical sequence in which they appear on the input partitioned data set.

Once a member of a data set has been found, no search is made for it on any subsequent input data set. Similarly, when all the selected members are found, the copy or load step is terminated even though all of the input data sets may not have been searched. For example, if members A and B are specified and A is found on the first of three input data sets, it is not searched for again; if B is found on the second input data set, the copy or load operation is successfully terminated after the second input data set has been processed, although both A and B may also exist on the third input data set.

However, if the first member name is not found on the first input data set, the search for that member stops and the first data set is searched for the second member. This process continues until the first input data set has been searched for all specified members. All the members that were found on the input data set are then processed for copying, unloading, or loading to the output data set. This process is repeated for the second input data set (except that the members that were found on the first input data set are not searched for again).

Note: Only one data set can be processed if an unload operation is to be performed. Multiple unload operations are allowed per job step; multiple INDD statements are <u>not</u> allowed per unload operation.

Copying Members That Have Alias Names

When copying members that have alias names, note the following:

- When the main member and its alias names are copied, they exist on the output partitioned data set in the same relationship they had on the input partitioned data set.
- When members with alias names are copied using the SELECT or EXCLUDE member option, those alias names that are to be selected or excluded must be explicitly named.

The rules for replacing or renaming members apply to both aliases and members; no distinction is made between them. However, the replace (R) option (on the SELECT statement) does not apply to an unload operation.

REPLACING IDENTICALLY NAMED MEMBERS

In many copy and load operations, the output partitioned data set may contain members that have names identical to the names of the input partitioned data set members to be copied or loaded. When this occurs, the user may specify that the identically named members are to be copied from the input partitioned data set to replace existing members.

The replace option allows an input member to override an existing member on the output partitioned data set with the same name. The pointer in the output partitioned data set directory is changed to point to the copied or loaded member.

If the replace option is not specified, input members are not copied when they have the same name as a member on the output partitioned data set.

The replace option can be specified on the data set or member level. This level is specified on a utility control statement.

When replace (R) is specified on the data set level with a COPY or INDD statement, the input data is processed as follows:

- In a full copy or load process, all members on an input partitioned data set are copied to an output partitioned data set; members whose names already exist on the output partitioned data set are replaced by the members copied or loaded from the input partitioned data set.
- In a selective copy or load process, all selected input members will be copied to the output data set, replacing any identically named output data set members.
- In an exclusive copy process, all nonexcluded members on input partitioned data sets are copied or loaded to an output partitioned data set replacing those duplicate named members on the output partitioned data set.

When replace is specified on the member level (specified as R on a SELECT statement), only selected members for which replace is specified are copied or loaded, and identically named members on the output partitioned data set are replaced.

There are differences between full, selective, and exclusive copy or load processing. These differences should be remembered when specifying the replace option and all of the output data sets contain member names common to some or all of the input partitioned data sets being copied or loaded. These differences are:

- When a full copy or load is performed, the output partitioned data set contains the replacing members that were on the last input partitioned data set copied.

- When a selective copy or load is performed, the output partitioned data set contains the selected replacing members that were found on the earliest input partitioned data set searched. Once a selected member is found, it is not searched for again; therefore, once found, a selected member is copied or loaded. If the same member exists on another input partitioned data set, it is not searched for, and hence, not copied or loaded.

- When an exclusive copy or load is performed, the output partitioned data set contains all members, except those specified for exclusion, that were on the last input partitioned data set copied or loaded.

REPLACING SELECTED MEMBERS

The user may specify the replace (R) option on either the data set or the member level when members are being selected for copying or loading.

If the replace option is specified on the data set level, all selected members found on the designated input data sets replace identically named members on the output partitioned data set. This is limited by the fact that once a selected member is found it is not searched for again.

If the replace option is specified on the member level, the specified members on the input data set replace identically named members on the output partitioned data set. Once a member is found it is not searched for again. (See "Replacing Identically Named Members" on page 41.)

RENAMING SELECTED MEMBERS

Selected members on input data sets can be copied and renamed on the output data set; the input and output data sets must not be the same. However, in the case of a copy or load operation, if the new name is identical to a member name on the output data set, the input member is not copied or loaded unless the replace option is also specified. See "SELECT Statement" on page 54for information on renaming selected members.

Renaming is not physically done to the input data set directory entry. The output data set directory, however, will contain the new name.

EXCLUDING MEMBERS FROM A COPY OPERATION

Members from one or more input data sets can be excluded from a copy, unload, or load operation. The excluded member is searched for on every input data set in the copy, unload, or load operation and is always omitted. Members are excluded from the input data sets named on an INDD statement that precedes the EXCLUDE statement. (See "COPY Statement" on page 51 and"EXCLUDE Statement" on page 56.)

The replace option can be specified on the data set level in an exclusive copy or load, in which case, nonexcluded members on the input data set replace identically named members on the output data set. See "Replacing Identically Named Members" on page 41for more information on the replace option.

COMPRESSING A DATA SET

A compressed data set is one that does not contain embedded, unused space. After copying or loading one or more input partitioned data sets to a new output partitioned data set (by means of a selective, exclusive, or full copy or load that does not involve replacing members), the output partitioned data set contains no embedded, unused space.

To make unused space available, either the entire data set must be scratched or it must be compressed in place. A compressed version can be created by specifying the same data set for both the input and the output parameters in a full copy step. A backup copy of the partitioned data set to be compressed in place should be kept until successful completion of an in-place compression is indicated (by an end-of-job message and a return code of 00).

An in-place compression does not release extents assigned to the data set. Inclusion, exclusion, or renaming of selected members cannot be done during the compression of a partitioned data set.

When the same ddname is specified for the INDD and OUTDD keywords (see "COPY Statement" on page 51) and the DD statement specifies a block size different from the block size specified in the DSCB, the DSCB block size is overridden; however, no physical reblocking or deblocking is performed by IEBCOPY. For information on reblocking load modules, see "Copying and Reblocking Load Modules."

MERGING DATA SETS

A merged data set is one to which an additional member is copied or loaded. It is created by copying or loading the additional members to an existing output partitioned data set; the merge operation—the ordering of the output partitioned data set's directory—is automatically performed by IEBCOPY.

If there is a question about whether or not enough directory blocks are allocated to the output partitioned data set to which an input data set is being merged, the output partitioned data set should be re-created with additional directory space prior to the merge operation.

RE-CREATING A DATA SET

A data set can be re-created by copying or loading it and allocating a larger amount of space than was allocated for the original data set. This application of IEBCOPY is especially useful if insufficient directory space was allocated to a data set. Space cannot be allocated in this manner for an existing partitioned data set into which members are being merged.

ALTERING LOAD MODULES IN PLACE

IEBCOPY can be used to alter load modules in place. Alter-in-place reads modules written by earlier runs of the linkage editor and inserts new relocation dictionary (RLD) counts. For modules copied by a program other than the linkage editor or IEBCOPY, alter-in-place can replace an erroneous RLD count by correcting PDS directory entries and control records. For more information, see "Inserting RLD Counts" on page 45.

Only members of a partitioned data set may be altered.

For the procedure used to invoke the alter-in-place function, see "ALTERMOD Statement" on page 53.

COPYING AND REBLOCKING LOAD MODULES

IEBCOPY can be used to copy and reblock load modules in a data set library. Copy/reblock copies a sequential (unloaded) data set or selected members from a partitioned data set onto a new or existing output partitioned data set. The text records, RLD, and control records are rebuilt; all other records are copied unchanged. For a description of how the RLD count is inserted, see "Inserting RLD Counts" on page 45.

The reblock function allows you to specify:

- A new maximum block size for compatibility with other systems or programs
- A minimum block size to improve DASD track utilization. The minimum block size specifies the smallest block which should be written on the end of a track.

The load modules will be blocked such that they can be re-link-edited and/or loaded by the loader, with the ability to include the whole module or only the indicated CSECTs.

Load libraries may be copied to devices with a larger or smaller block size than the input block size.

IEBCOPY will determine the amount of space remaining on a track before assigning a new block size, and if this amount is less than the output block size, it will attempt to determine whether a smaller block can be written to utilize the remaining space on the track.

The maximum block size which can be handled by the linkage editor is 18K.

For the procedure used to copy and reblock load modules, see "COPYMOD Statement" on page 53.

LOAD MODULE REQUIREMENTS

IEBCOPY requires that the members of the input data set which are to be altered or copied/reblocked must qualify as **load modules**; that is, they must possess characteristics such that they can be loaded by the system fetch routine (IEWFETCH) or re-link-edited by the linkage editor. Members which are not recognized as load modules will be unaffected by the alter-in-place or copy/reblock operation.

Load modules in either overlay or scatter-load format and modules which were link-edited with the noneditable (NE) attribute or with an assigned origin other than zero cannot be altered in place. For more information on module format and attributes, see Linkage Editor and Loader.

The PDS directory entry for a load module must meet the following requirements:

1. The entry must be at least 34 bytes long (standard length for entries is only 12 bytes).
2. Bytes 26 and 27 must contain the length of the first text record, and this length must be equal to the length specified by the first control record.

Any record in a load module which precedes the first control record must be one of the following:

- A symbol record (SYM)
- A composite external symbol dictionary record (CESD)
- An external symbol dictionary record (ESD)
- A scatter/translation record (STT)
- A CSECT identification record (IDR)

RLD and control records must be:

- An RLD record: '0000 xx10'B in byte 1,
- A control record: '0000 xx01'B in byte 1,

- An RLD and control record: '0000 xx11'B in byte 1, or
- The length specified by the value in bytes 5-6 plus the value in bytes 7-8 plus 16. Control records must contain the length of the following text record in bytes 15-16.

The sequence of records following a control or RLD/control record must be:

- Text, End-of-Module/End-of-Segment,
- Text, RLD, End-of-Module/End-of-Segment,
- Text, RLD/control,
- Text, RLD, (RLD, . . .), End-of-Module/End-of-Segment, or
- Text, RLD, (RLD, . . .), RLD/control.

INSERTING RLD COUNTS

Each block of text in a load module is preceded by a control record and may be followed by one or more RLD and/or control records. These records are variable length with a maximum of 256 bytes. They may contain only RLD data or only control data or both RLD and control data.

The term 'number' or 'count' of RLD records is used to mean the number of these records containing RLD data/control data which follow a block of text in a module library.

The system fetch routine (IEWFETCH) executes fewer start I/O instructions if the number of these records following a block of text is known. The number of RLD records following each block of text is inserted into the control record which immediately precedes that block of text. In addition, the number of RLD records which follow the first block of text for a load module is inserted into the PDS directory entry for that module.

The linkage editor inserts RLD counts in the control records and in the PDS directory entries.

INPUT AND OUTPUT

IEBCOPY uses the following input:

- An input data set that contains the members to be copied, loaded, merged, altered, reblocked, or unloaded to a sequential data set.
- A control data set that contains utility control statements. The control data set is required for a copy, unload, load, or merge operation.

IEBCOPY does not support VIO (virtual I/O) data sets.

IEBCOPY produces the following output:

- An output data set, which contains the copied, merged, altered, reblocked, unloaded, or loaded data. The output data set is either a new data set (from a copy, reblock, load, or unload) or an old data set (from a merge, compress-in-place, copy, alter, or load).
- A message data set, which contains informational messages (for example, the names of copied, unloaded, or loaded members) and error messages, if applicable.
- Spill data sets, which are temporary data sets used to provide space when not enough virtual storage is available for the input and/or output partitioned data set directories. These data sets are opened only when needed.

RETURN CODES

IEBCOPY returns a code in register 15 to indicate the results of program execution. The return codes and their meanings are listed below.

Codes	Meaning
00 (00 hex)	Successful completion.
04 (04)	A condition exists from which recovery may be possible.
08 (08)	An unrecoverable error exists. The job step is terminated.

Figure 17. IEBCOPY Return Codes

CONTROL

IEBCOPY is controlled by job control statements and utility control statements.

JOB CONTROL STATEMENTS

Figure 18 on page 47 shows the job control statements for IEBCOPY.

PARM Information on the EXEC Statement

The EXEC statement for IEBCOPY can contain PARM information that is used to define the number of bytes used as a buffer. The PARM parameter can be coded:

PARM='SIZE=nnnnnnnn**[K]'**

The nnnnnnnn can be replaced by 1 to 8 decimal digits. The K causes the nnnnnnnn to be multiplied by 1024 bytes.

If PARM is not specified, or a value below the minimum buffer size is specified, IEBCOPY defaults to the minimum. Minimum buffer size is twice the maximum of the input or output block sizes or four times the input or output track capacities, whichever is larger.

The maximum buffer size that can be specified is equal to the storage remaining in the storage area gotten when IEBCOPY issues a conditional one-megabyte storage request (GETMAIN) for work areas and buffers. If the value specified in PARM exceeds this maximum, IEBCOPY defaults to the maximum.

A request for too much buffer storage may result in increased system paging because of a lack of available system page frames. This will degrade overall system performance.

SYSPRINT DD Statement

The SYSPRINT DD statement is required and must define a data set with fixed blocked or fixed records. The block size for the SYSPRINT data set must be a multiple of 121. Any blocking factor may be specified, with a maximum allowable block size of 32767 bytes.

Statement	Use
JOB	Initiates the job.
EXEC	Specifies the program name (PGM=IEBCOPY) or, if the job control statements reside in the procedure library, the procedure name. This statement can include optional PARM information to define the size of the buffer to be used; see "PARM Information on the EXEC Statement."
SYSPRINT DD	Defines the sequential message data set used for listing statements and messages. This data set can be written to a system output device, a tape volume, or a direct access volume.
anyname1 DD	Defines an input partitioned data set. These DD statements can describe partitioned data sets on direct access devices or sequential data sets, created as a result of unload operations, on tape or direct access devices. The data set can be defined by a data set name, as a cataloged data set, or as a data set passed from a previous job step.
anyname2 DD	Defines an output partitioned data set. These DD statements can describe partitioned data sets on direct access devices or sequential data sets, created as a result of unload operations, on tape or direct access devices.
SYSUT3 DD	Defines a spill data set on a direct access device. SYSUT3 is used when there is no space in virtual storage for some or all of the current input partitioned data set's directory entries. SYSUT3 may also be used when not enough space is available in virtual storage for retaining information during table sorting.
SYSUT4 DD	Defines a spill data set on a direct access device. SYSUT4 is used when there is no space in virtual storage for the current output partitioned data set's merged directory and the output partitioned data set is not new.
SYSIN DD	Defines the control data set. The control data set normally resides in the input stream; however, it can reside on a system input device, a tape volume, or a direct access volume.

Figure 18. Job Control Statements for IEBCOPY

anyname1 and anyname2 DD Statements

DD statements are required for input and output data sets. There must be one DD statement for each unique data set used for input and one DD statement for each unique data set used for output in the job step. For an unload operation, only one input data set may be specified for each output data set.

Data sets used as input data sets in one copy operation can be used as output data sets in another copy operation, and vice versa.

Input data sets cannot be concatenated. The maximum block size for input data sets to be unloaded is 32767.

SYSIN DD Statement

The SYSIN DD statement is required and must define a data set with fixed block or fixed records. The block size for the SYSIN data set must be a multiple of 80. Any blocking factor may be specified, with a maximum allowable block size of 32767 bytes.

IEBCOPY UNLOADED DATA SET BLOCK SIZE

The block size for unloaded data sets is determined by the following steps:

1. The minimum block size for the unloaded data set is calculated as being equal to the larger of:
   - 284 bytes, or
   - 20 bytes + the block size and key length of the input data set.
2. If a user-supplied block size was specified, and it is larger than 284 bytes, it will be passed to step 3. Otherwise, the minimum size is passed.
3. The block size value passed from step 2 is then compared with the largest block size acceptable to the output device. If the output device capacity is less than the block size passed in step 2, the unloaded data set block size is set to the maximum allowed for the output device.
4. The logical record length (LRECL) is then set to the minimum block size calculated in step 1 minus 4 bytes.
5. The block size is stored in the first control record (COPYR1) and used at load time. Block size of the unloaded data set must not be changed before the data set is loaded. Be sure to specify the desired block size at unload time if it is other than that taken by default as indicated above.

For unload and load operations, requests are handled in the same way as for a copy operation.

Fixed or variable records can be reblocked. Reblocking or deblocking is done if the block size of the input partitioned data set is not equal to the block size of the output partitioned data set.

An unloaded partitioned data set will have a variable spanned record format. When an unloaded data set is subsequently loaded, the output data set will have the same characteristics it had before the unload operation, unless specified differently by the user.

Figure 19 shows how input record formats can be changed. In addition, any record format can be changed to the undefined format (in terms of its description in the DSCB).

Input	Output
Fixed	Fixed-Blocked
Fixed-Blocked	Fixed
Variable	Variable-Blocked
Variable-Blocked	Variable

Figure 19. Changing Input Record Format Using IEBCOPY

System data sets should not be compressed in place unless the subject partitioned data set is made **nonsharable**. The libraries in which IEBCOPY resides (SYS1.LINKLIB and SYS1.SVCLIB) must not be compressed by IEBCOPY unless IEBCOPY is first transferred to a JOBLIB.

Space Allocation

Sometimes it is necessary to allocate space on spill data sets (SYSUT3 and SYSUT4). The space to be allocated for SYSUT3 depends on the number of members to be copied or loaded. The space to be allocated for SYSUT4 depends on the number of directory blocks to be written to the output data set.

To conserve space on the direct access volume, an initial quantity and a secondary quantity for space allocation may be used, as shown in the following SPACE parameter:

```
SPACE=(c,(x,y))
```

The c value should be a block length of 80 for SYSUT3 and of 256 for SYSUT4. The x value is the number of blocks in the primary allocation, and the y value is the number of blocks in the secondary allocation.

For SYSUT3, x + 15y must be equal to or greater than the number of members in the largest input partitioned data set in the copy operation, multiplied by 1.05.

For SYSUT4, x + 15y must be equal to or greater than the number of blocks allocated to the largest output partitioned data set directory in the IEBCOPY job step.

For example, if there are 700 members on the largest input partitioned data set, space could be allocated for SYSUT3 as follows:

```
SPACE=(80,(60,45))
```

However, the total amount of space required for SYSUT3 in the worst case is used only if needed. If space is allocated in this manner for SYSUT4, the user must specify in his SYSUT4 DD statement:

```
DCB=(KEYLEN=8)
```

IEBCOPY ignores all other DCB information specified for SYSUT3 and/or SYSUT4. Multivolume SYSUT3 and SYSUT4 data sets are not supported.

The temporary spill data sets may or may not be opened, depending on the amount of virtual storage available; therefore, the SYSUT3 and SYSUT4 DD statements should always appear in the job stream.

Refer to Data Management Services for more information on estimating data set space allocations.

Restrictions

- IEBCOPY must run from an authorized library because of special storage key requirements for IEBCOPY I/O appendages.
- Variable block spanned format data sets are not supported.
- VIO is not supported by IEBCOPY for SYSUT4, nor for partitioned input or output data sets.
- When merging into or compressing system libraries, do not specify DISP=SHR. The results of a merge into or compress of the current SYS1.LINKLIB or SYS1.SVCLIB would be unpredictable.

- IEBCOPY does its own buffering; therefore, coding the BUFNO parameter in the DCB will cause a JCL error.
- Reblocking or deblocking cannot be done if either the input or the output data set has undefined format records, keyed records, track overflow records, note lists, or user TTRNs, or if compress-in-place is specified. Load modules, however, may be reblocked using the COPYMOD statement.

The compress-in-place function cannot be performed for the following:

- An unloaded data set
- A data set with track overflow records
- A data set with keyed records
- A data set for which reblocking is specified in the DCB parameter
- An unmovable data set

Note: If IEBCOPY creates a copied library (partitioned data set) whose block size is smaller than the logical record length of the original library, a return code of 4 is issued, with message IEB175I. If IEBCOPY is used later to compress-in-place the output library, the operation will fail and this library becomes unusable.

UTILITY CONTROL STATEMENTS

IEBCOPY is controlled by the following utility control statements:

Statement	Use
COPY	Indicates the beginning of a COPY operation.
ALTERMOD	Specifies the load module(s) to be altered in place.
COPYMOD	Specifies the load module(s) to be copied and reblocked.
SELECT	Specifies which members in the input data set are to be copied.
EXCLUDE	Specifies members in the input data set to be excluded from the copy step.

Figure 20. IEBCOPY Utility Control Statements

In addition, when INDD, a COPY statement parameter, appears on a card other than the COPY statement, it is referred to as an INDD statement; it can function as a control statement in this context.

Continuation requirements for utility control statements are described in "Continuing Utility Control Statements" on page 5.

COPY Statement

The COPY statement is required to initiate one or more IEBCOPY copy, unload, or load operations. Any number of operations can follow a single COPY statement; any number of COPY statements can appear within a single job step.

IEBCOPY uses a copy operation/copy step concept.[1] A copy operation starts with a COPY statement and continues until either another COPY statement or a COPYMOD or ALTERMOD statement is found, or the end of the control data set is found. Within each copy operation, one or more copy steps are present. Any INDD statement directly following a SELECT or EXCLUDE statement marks the beginning of the next copy step and the end of the preceding copy step within the copy operation. If such an INDD statement cannot be found in the copy operation, then the copy operation consists of only one copy step.

Figure 21 shows the copy operation/copy step concept. Two copy operations are shown in the figure: the first begins with the statement containing the name COPOPER1, and the second begins with the statement containing the name COPOPER2.

First Copy Operation

```
STEP 1      COPOPER1        COPY     OUTDD=AA,INDD=ZZ
                                     INDD=(BB,CC)
                                     INDD=DD
                                     INDD=EE
                            SELECT   MEMBER=(MEMA,MEMB)
                            SELECT   MEMBER=MEMC

STEP 2                               INDD=GG
                                     INDD=HH
                            EXCLUDE  MEMBER=(MEMD,MEMH)
```

Second Copy Operation

```
STEP 1      COPOPER2        COPY     OUTDD=YY,I=(MM,PP),LIST=NO
                            SELECT   MEMBER=MEMB

STEP 2                               INDD=KK
                                     INDD=(LL,NN)
            /*
```

Figure 21. Multiple Copy Operations within a Job Step

There are two copy steps within the first copy operation shown in Figure 21: the first begins with the COPY statement and continues through the two SELECT statements; the second begins with the first INDD statement following the two SELECT statements and continues through the EXCLUDE statement preceding the second COPY statement. There are two copy steps within the second copy operation: the first begins with the COPY statement and continues through the SELECT statement; the second begins with the INDD statement immediately following the SELECT statement and ends with the same /* (delimiter) statement that ended the copy operation.

[1] The same applies to an unload or load operation or step.

The format of the COPY statement is:

[label]	COPY	OUTDD=ddname INDD=[(]ddname1[,ddname2] [,(ddname3,R)][,...][)] [,LIST=NO]

The control statement operation and keyword parameters can be abbreviated to their initial letters; for example, COPY can be abbreviated to C and OUTDD can be abbreviated to O.

If there are no keywords other than OUTDD on the COPY card, compatibility with the previous version of the data set is implied. In this case, comments may not be placed on this card.

The OUTDD and INDD keyword parameters on COPY statements name DD statements that define data sets to be copied, unloaded, or loaded. The INDD parameter names the DD statement that identifies the input data set. The OUTDD parameter names the DD statement that identifies the output data set.

Only one INDD and one OUTDD keyword may be placed on a single card. OUTDD must appear on the COPY statement. When INDD appears on a separate card, no other operands may be specified on that card. If INDD appears on a separate card, it is not preceded by a comma.

The characteristics of the input and output data sets depend on the operation to be performed, as follows:

- If a data set is to be copied, the input and output data sets must both be partitioned data sets.
- If a data set is to be loaded, the input data set may be either partitioned or sequential; the output data set must be partitioned.
- If a data set is to be unloaded, the input data set must be either a partitioned data set or a sequential data set that was created as a result of a previous unload operation. The output data set may reside on either a direct access or tape volume. If the output data set is to reside on a direct access volume, the organization of the data set must be specified as sequential. To specify sequential organization for a direct access data set, specify the SPACE parameter, omitting the directory or index value.

If more than one ddname is specified, the input partitioned data sets are processed in the same sequence as that in which the ddnames are specified.

A COPY statement must precede a SELECT or EXCLUDE statement when members are selected for or excluded from a copy, unload, or load step. In addition, if an input ddname is specified on a separate INDD statement, it must follow the COPY statement and precede the SELECT or EXCLUDE statement to which it applies. If one or more INDD statements are immediately followed by the /* card or another COPY or COPYMOD or ALTERMOD statement, a full copy, unload, or load is invoked onto the most recent previously specified output partitioned data set.

A full copy, unload, or load is invoked only by specifying different input and output ddnames; that is, by omitting the SELECT or EXCLUDE statement from the copy step.

The compress-in-place function is valid for partitioned data sets. Compress-in-place is normally invoked by specifying the same ddname for both the OUTDD and INDD parameters of a COPY statement. If multiple entries are made on the INDD statement, a compress-in-place will occur if one of the input ddnames is the same as the ddname specified by the OUTDD parameter of the COPY statement, provided that SELECT or EXCLUDE is not specified.

When a compression is invoked by specifying the same ddname for the INDD and OUTDD parameters, and the DD statement specifies a block size that differs from the block size specified in the DSCB, the DSCB block size is overridden; however, no physical reblocking or deblocking is done by IEBCOPY.

ALTERMOD Statement

The ALTERMOD statement is required to alter load modules in place. The function is designed to read modules which were written by earlier versions of the linkage editor and to insert RLD counts. It can also be used to alter modules which may have an erroneous RLD count—for example, modules which were copied by a program other than the linkage editor or IEBCOPY.

Only PDS directory entries and control records will be modified. If the control records are already correct, they will not be rewritten.

Members which are not recognized as load modules will not be altered.

Load modules in either overlay or scatter-load format and modules which were link-edited with the noneditable (NE) attribute or with an assigned origin other than zero will not be altered.

The alter-in-place function may be performed multiple times for the same load module or module library. Altering has no cumulative effect.

The format of the ALTERMOD statement is:

[label]	**ALTERMOD**	**OUTDD=**ddname **[,LIST=NO]**

OUTDD specifies the partitioned data set which is to be altered.

The replace (R) and RENAME functions of IEBCOPY cannot be specified in the same step with ALTERMOD.

COPYMOD Statement

The COPYMOD statement is required to copy, reblock, and alter modules in a library. When copying load modules, the selected members will be copied from the input data set(s) to the output data set. The output data set may be new or it may be an existing load library to which members are to be added. The output data set must be a partitioned data set, and it cannot also be an input data set (reblock-in-place is not permitted).

The text records and the RLD/control records will be rebuilt. Other records such as SYM and CESD records will be copied unchanged.

Load modules in either overlay or scatter-load format and modules which were link-edited with the noneditable (NE) attribute or with an assigned origin other than zero will be copied, but not reblocked or altered (that is, as if the member was specified with a COPY statement). Members which are not recognized as load modules will be copied, but not reblocked or altered.

Note that modules which are not reblocked by COPYMOD cannot be copied to a device which has a track size less than the input block size. They may, however, be re-link-edited with a smaller block size.

The replace (R) function may be specified with input ddnames and/or member names to cause like-named modules to be replaced, or it may be omitted to prevent the copying of like-named modules.

The rename function may be invoked to specify a new name for the selected member. For more information, see "SELECT Statement."

IEBCOPY can unload modules to a sequential data set via the COPY function, and the output of that step can be input to a subsequent COPYMOD step in which the output data set is the same as the input to the unload step. This would also provide a backup copy in the sequential data set.

The format of the COPYMOD statement is:

[label]	**COPYMOD**	**OUTDD=**ddname **,INDD=[(]**ddname1**[,**ddname2**]** **[,(**ddname3**,R)][,...][)]** **[,MAXBLK={**nnnnn**|**nn**K}]** **[,MINBLK={**nnnnn**|**nn**K}]** **[,LIST=NO]**

INDD specifies the partitioned or sequential (unloaded) data set from which load modules are to be read. **OUTDD** specifies the partitioned data set to which load modules are to be copied. **MAXBLK** specifies the maximum block size for records in the output data set. **MINBLK** specifies the minimum block size for records in the output data set.

SELECT Statement

The SELECT statement specifies members (or modules, in the case of ALTERMOD or COPYMOD) to be selected from input data sets to be altered, copied, loaded, or unloaded to an output data set. This statement is also used to rename and/or replace selected members on the output data set. More than one SELECT statement may be used in succession, in which case the second and subsequent statements are treated as a continuation of the first.

The SELECT statement must follow either a COPY statement that includes an INDD parameter, a COPYMOD statement, or one or more INDD statements. A SELECT statement cannot appear with an EXCLUDE statement in the same copy, unload, or load step, and it cannot be used with a compress-in-place function.

When a selected member is found on an input data set, it is not searched for again, regardless of whether the member is copied, unloaded, or loaded. A selected member will not replace an identically named member on the output partitioned data set unless the replace option is specified on either the data set or member level. (For a description of replacing identically named

members, see "Replacing Identically Named Members" on page 41 and "Replacing Selected Members" on page 42.) In addition, unless the replace option is specified, a renamed member will not replace a member on the output partitioned data set that has the same new name as the renamed member.

The replace (R) and rename (newname) options cannot be specified with ALTERMOD.

The format of the SELECT statement is:

[label]	SELECT	MEMBER= {[([]name1[,name2][,...][])]| {([(name1,newname[,R])[,...]| (name1,newname)[,...]| (name1,,R)[,...][])]}

where:

MEMBER=
specifies the members to be selected from the input data set. The values that can be coded are:

name
specifies the name of a member that is to be selected in a copy step. Each member name specified within one copy step must be unique; that is, duplicate names cannot be specified as either old names, or new names, or both, under any circumstances.

newname
specifies a new name for a selected member. The member is copied, unloaded, or loaded to the output partitioned data set using its new name. If the name already appears on the output partitioned data set, the member is not copied unless replacement (R) is also specified. newname cannot be specified with ALTERMOD.

R
specifies that the input member is to replace any identically named member that exists on the output partitioned data set. The replace option is not valid for an unload operation. R cannot be specified with ALTERMOD.

The control statement operation and keyword parameters can be abbreviated to their initial letters; SELECT can be abbreviated to S and MEMBER can be abbreviated to M.

To rename a member, the old member name is specified in the SELECT statement, followed by the new name and, optionally, the R parameter. When this option is specified, the **old** member name and **new** member name must be enclosed in parentheses. When any option within parentheses is specified anywhere in the MEMBER field, the entire field, exclusive of the MEMBER keyword, must be enclosed in a second set of parentheses.

EXCLUDE Statement

The EXCLUDE statement specifies members to be excluded from the copy, unload, or load step. Unlike the selective copy/alter/unload/load, an exclusive copy/alter/unload/load causes all members (or modules, in the case of ALTERMOD or COPYMOD) specified on each EXCLUDE statement to be omitted from the operation.

More than one EXCLUDE statement may be used in succession, in which case the second and subsequent statements are treated as a continuation of the first. The EXCLUDE statement must follow either a COPY statement that includes an INDD parameter, an ALTERMOD or COPYMOD statement, or one or more INDD statements. An EXCLUDE statement cannot appear with a SELECT statement in the same copy, unload, or load step; however, both may be used following a COPY statement for a copy or load operation. The EXCLUDE statement cannot be used with a compress-in-place function.

The format of the EXCLUDE statement is:

[label]	**EXCLUDE**	**MEMBER=[(]**membername1**[,**membername2**]...[)]**

The control statement operation and keyword parameters can be abbreviated to their initial letters; EXCLUDE can be abbreviated to E and MEMBER can be abbreviated to M.

If neither SELECT nor EXCLUDE is specified, the entire data set is copied (a "full copy").

Parameters	Applicable Control Statements	Description of Parameters
INDD	COPY COPYMOD	**INDD=[(]**ddname1**[**,ddname2**][**,(ddname3,R**)][**,...**][)]** specifies the names of the input partitioned data sets. INDD may, optionally, be placed on a separate line following a COPYMOD or COPY statement containing the OUTDD parameter, another INDD statement, a SELECT statement, or an EXCLUDE statement. These values can be coded: ddname specifies the ddname, which is specified on a DD statement, of an input data set. In the case of COPYMOD, this is the name of a load module. For an unload operation, only one ddname may be specified per COPY statement. If more than one ddname is specified in the case of a copy or load operation, the input data sets are processed in the same sequence as the ddnames are specified. **R** specifies that all members to be copied or loaded from this input data set are to replace any identically named members on the output partitioned data set. (In addition, members whose names are not on the output partitioned data set are copied or loaded as usual.) When this option is specified with the INDD parameter, it does not have to appear with the MEMBER parameter (discussed in "SELECT Statement" on page 54) in a selective copy operation. When this option is specified, the ddname and the R parameter must be enclosed in a set of parentheses; if it is specified with more than one ddname in INDD, the entire field, exclusive of the INDD parameter, must be enclosed in a second set of parentheses.
LIST	COPY COPYMOD ALTERMOD	**LIST=NO** specifies that the names of copied members are not to be listed on SYSPRINT at the end of each input data set. **Default:** The names of copied members are listed.

Parameters	Applicable Control Statements	Description of Parameters
MAXBLK	COPYMOD	**MAXBLK={nnnnn|nnK}** specifies the maximum block size for records in the output partitioned data set. MAXBLK is normally used to specify a smaller block size than the default, in order to make the records in the data set compatible with other systems or programs. nnnnn is specified as a decimal number; K indicates that the nn value is multiplied by 1024 bytes. MAXBLK may be specified with or without MINBLK. **Default:** The track size for the output device or 18K, whichever is smaller. If a value greater than 18K (18432) or less than 4K (4096) is specified, the default is used.
MEMBER	SELECT	**MEMBER={[(]name1[,name2][,...][)]|** **{([name1,newname[,R.])[,...]|** **(name1,newname)[,...]|** **(name1,,R)[,...]})}** specifies the members to be selected from the input data set. The values that can be coded for SELECT are: name specifies the name of a member that is to be selected in a copy step. Each member name specified within one copy step must be unique; that is, duplicate names cannot be specified as either old names, or new names, or both, under any circumstances. If no member name is specified, the entire data set is included in the operation. newname specifies a new name for a selected member. The member is copied, unloaded, or loaded to the output partitioned data set using its new name. If the name already appears on the output partitioned data set, the member is not copied unless replacement (R) is also specified. R specifies that the input member is to replace any identically named member that exists on the output partitioned data set. The replace option is not valid for an unload or alter operation.

Parameters	Applicable Control Statements	Description of Parameters
MEMBER	EXCLUDE	**MEMBER=[(]**membername1**[**,membername2**]...[)]** specifies members on the input data set that are not to be copied, unloaded, or loaded to the output data set. The members are not deleted from the input data set unless the entire data set is deleted. (This can be done by specifying DISP=DELETE in the operand field of the input DD job control statement.) Each member name specified within one copy step must be unique.
MINBLK	COPYMOD	**MINBLK={**nnnnn**|**nn**K}** specifies the minimum block size for records in the output partitioned data set. MINBLK specifies the smallest block which should be written on the end of a track for the purpose of improving utilization of DASD storage. A small MINBLK value will improve track utilization; however, a large MINBLK value (close to the track size) will improve system fetch (IEWFETCH) performance. When determining the value of MINBLK, you should consider the importance of fetch performance versus optimal DASD storage. In any case, in order to have room for RLD counts, the value of MINBLK should be less than the size of one full track. nnnnn is specified as a decimal number; K indicates that the nn value is multiplied by 1024 bytes. MINBLK may be specified with or without MAXBLK. **Default:** 1K (1024). If a value greater than MAXBLK or less than 1K is specified, 1K is used. The default for the installation can be changed by altering the value in the assembler statement 'MINBLK DC F'1024'' in the macro IEBMCA and reassembling the module IEBDSCPY.
OUTDD	COPY COPYMOD ALTERMOD	**OUTDD=** ddname specifies the name of the output partitioned data set. One ddname is required for each copy, unload, or load operation; the ddname used must be specified on a DD statement. When the COPY or COPYMOD or ALTERMOD statement is used, OUTDD must be specified.

IEBCOPY EXAMPLES

The following examples illustrate some of the uses of IEBCOPY. Figure 22 on page 61 can be used as a quick-reference guide to IEBCOPY examples. The numbers in the "Example" column point to examples that follow.

Examples that use **disk** or **tape**, in place of actual device numbers, must be changed before use. See "DASD and Tape Device Support" on page 3 for valid device number notation.

Operation	Device	Comments	Example
COPY	Disk	Full Copy. The input and output data sets are partitioned.	1
COPY	Disk	Multiple input partitioned data sets. Fixed-blocked and fixed-record formats.	2
COPY	Disk	All members are to be copied. Identically named members on the output data set are to be replaced. The input and output data sets are partitioned.	3
COPY	Disk	Selected members are to be copied. Variable-blocked data set is to be created. Record formats are variable-blocked and variable. The input and output data sets are partitioned.	4
COPY	Disk	Selected members are to be copied. One member is to replace an identically named member on the output data set. The input and output data sets are partitioned.	5
COPY	Disk	Selected members are to be copied. Members found on the first input data set replace identically named members on the output data set. The input and output data sets are partitioned.	6
COPY	Disk	Selected members are to be copied. Two members are to be renamed. One renamed member is to replace an identically named member on the output data set. The input and output data sets are partitioned.	7
COPY	Disk	Exclusive Copy. Fixed-blocked and fixed-record formats. The input and output data sets are partitioned.	8
Unload and Compress-in-place	Disk and Tape	Copy a partitioned data set to tape (unload) and compress-in-place if the first step is successful.	9
COPY and Compress-in-place	Disk	Full copy to be followed by a compress-in-place of the output data set. Replace specified for one input data set. The input and output data sets are partitioned.	10
COPY	Disks	Multiple copy operations. The input and output data sets are partitioned.	11
COPY	Disks	Multiple copy operations.	12
Unload	Disk and Tape	A partitioned data set is to be unloaded to tape.	13
Load	Tape and Disk	An unloaded data set is to be loaded to disk.	14
Unload, Load, and COPY	Disk and Tape	Selected members are to be unloaded, loaded, and copied. The input data set is partitioned; the output data set is sequential.	15
Alter in Place	Disk	Selected members are to be altered in place.	16

Figure 22 (Part 1 of 2). IEBCOPY Example Directory

Operation	Device	Comments	Example
Copy, alter, and reblock	Disk	Selected members are copied to a new data set, altered, and reblocked to various sizes.	17
Copy, alter, and reblock	Disk and Tape	All members copied to tape; library scratched; members copied back to library, altered, and reblocked.	18

Figure 22 (Part 2 of 2). IEBCOPY Example Directory

IEBCOPY EXAMPLE 1

In this example, a partitioned data set (DATASET5) is copied from one disk volume to another. Figure 23 shows the input and output data sets before and after processing.

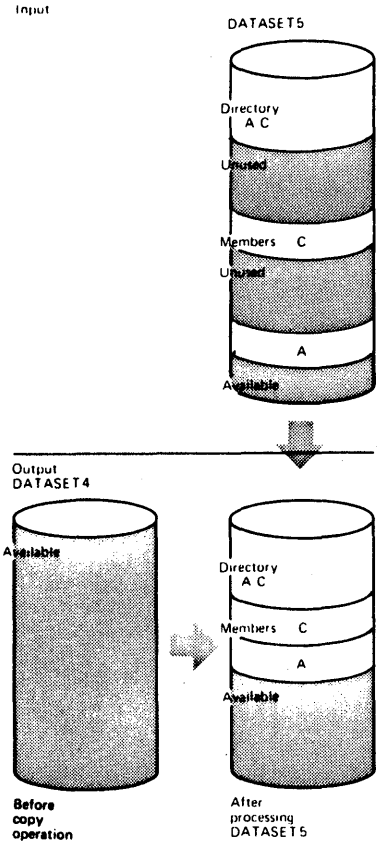


Figure 23. Copying a Partitioned Data Set—Full Copy

```
//COPY     JOB   ....
//JOBSTEP  EXEC  PGM=IEBCOPY
//SYSPRINT DD    SYSOUT=A
//INOUT4   DD    DSNAME=DATASET4,UNIT=3350,VOL=SER=111112,
//                DISP=(NEW,KEEP),SPACE=(TRK,(5,1,2))
//INOUT5   DD    DSNAME=DATASET5,UNIT=3350,VOL=SER=111113,
//                DISP=SHR
//SYSUT3   DD    UNIT=SYSDA,SPACE=(TRK,(1))
//SYSUT4   DD    UNIT=SYSDA,SPACE=(TRK,(1))
//SYSIN    DD    *
COPYOPER  COPY  OUTDD=INOUT4,INDD=INOUT5
/*
```

The control statements are discussed below:

- INOUT4 DD defines a new partitioned data set (DATASET4) that is to be kept after the copy operation. Five tracks are allocated for the data set on a 3350 volume. Two blocks are allocated for directory entries.
- INOUT5 DD defines a partitioned data set (DATASET5), that resides on a 3350 volume and contains two members (A and C).
- SYSUT3 DD defines a temporary spill data set. One track is allocated on a disk volume.
- SYSUT4 DD defines a temporary spill data set. One track is allocated on a disk volume.
- SYSIN DD defines the control data set, which follows in the input stream. The data set contains a COPY statement.
- COPY indicates the start of the copy operation. The absence of a SELECT or EXCLUDE statement causes a default to a full copy. The OUTDD parameter specifies INOUT4 as the DD statement for the output data set (DATASET4); the INDD parameter specifies INOUT5 as the DD statement for the input data set. After the copy operation is finished, the output data set (DATASET4) will contain the same members that are on the input data set (DATASET5); however, there will be no embedded, unused space on DATASET4.

The temporary spill data sets may or may not be opened, depending on the amount of virtual storage available; therefore, the SYSUT3 and SYSUT4 DD statements should always appear in the job stream.

IEBCOPY EXAMPLE 2

In this example, members are copied from three input partitioned data sets (DATASET1, DATASET5, and DATASET6) to an existing output partitioned data set (DATASET2). The sequence in which the control statements occur controls the manner and sequence in which partitioned data sets are processed. Figure 24 on page 64 shows the input and output data sets before and after processing.

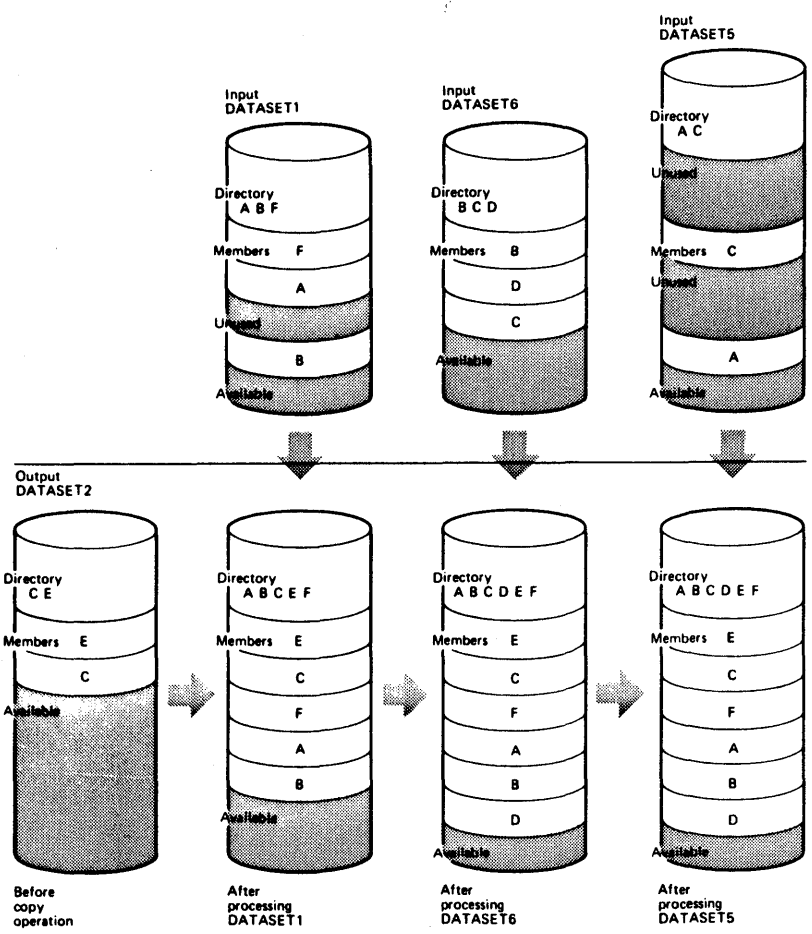


Figure 24. Copying from Three Input Partitioned Data Sets

```
//COPY     JOB   ....
//JOBSTEP  EXEC  PGM=IEBCOPY
//SYSPRINT DD    SYSOUT=A
//INOUT1   DD    DSNAME=DATASET1,UNIT=3330,VOL=SER=111112,
//              DISP=SHR
//INOUT5   DD    DSNAME=DATASET5,UNIT=3350,VOL=SER=111114,
//              DISP=OLD
//INOUT2   DD    DSNAME=DATASET2,UNIT=3350,VOL=SER=111115,
//              DISP=(OLD,KEEP)
//INOUT6   DD    DSNAME=DATASET6,UNIT=3350,VOL=SER=111117,
//              DISP=(OLD,DELETE)
//SYSUT3   DD    UNIT=SYSDA,SPACE=(TRK,(1))
//SYSUT4   DD    UNIT=SYSDA,SPACE=(TRK,(1))
//SYSIN    DD    *
COPYOPER  COPY   OUTDD=INOUT2
                 INDD=INOUT1
                 INDD=INOUT6
                 INDD=INOUT5
/*
```

The control statements are discussed below:

- INOUT1 DD defines a partitioned data set (DATASET1). This data set, which resides on a 3330 volume, contains three members (A, B, and F) in fixed format with a logical record length of 80 bytes and a block size of 80 bytes.
- INOUT5 DD defines a partitioned data set (DATASET5), which resides on a 3350 volume. This data set contains two members (A and C) in fixed blocked format with a logical record length of 80 bytes and a block size of 160 bytes.
- INOUT2 DD defines a partitioned data set (DATASET2), which resides on a 3350 volume. This data set contains two members (C and E) in fixed blocked format. The members have a logical record length of 80 bytes and a block size of 240 bytes.
- INOUT6 DD defines a partitioned data set (DATASET6), which resides on a 3350 volume. This data set contains three members (B, C, and D) in fixed-block format with a logical record length of 80 bytes and a block size of 400 bytes. This data set is to be deleted when processing is completed.
- SYSUT3 DD defines a temporary spill data set. One track is allocated on a disk volume.
- SYSUT4 DD defines a temporary spill data set. One track is allocated on a disk volume.
- SYSIN DD defines the control data set, which follows in the input stream. The data set contains a COPY statement and three INDD statements.
- COPY indicates the start of the copy operation. The absence of a SELECT or EXCLUDE statement causes a default to a full copy. The OUTDD parameter specifies INOUT2 as the DD statement for the output data set (DATASET2).
- The first INDD statement specifies INOUT1 as the DD statement for the first input data set (DATASET1) to be processed. All members (A, B, and F) are copied to the output data set (DATASET2).
- The second INDD statement specifies INOUT6 as the DD statement for the second input data set (DATASET6) to be processed. Processing occurs, as follows: (1) members B and C, which already exist on DATASET2, are not copied to the output data set (DATASET2), (2) member D is copied to the output data set (DATASET2), and (3) all members on DATASET6 are lost when the data set is deleted.
- The third INDD statement specifies INOUT5 as the DD statement for the third input data set (DATASET5) to be processed. No members are copied to the output data set (DATASET2) because all of them exist on DATASET2.

The temporary spill data sets may or may not be opened, depending on the amount of virtual storage available; therefore, the SYSUT3 and SYSUT4 DD statements should always appear in the job stream.

IEBCOPY EXAMPLE 3

In this example, members are copied from an input partitioned data set (DATASET6) to an existing output partitioned data set (DATASET2). In addition, all copied members replace identically named members on the output partitioned data set.

Figure 25 on page 66 shows the input and output data sets before and after processing.

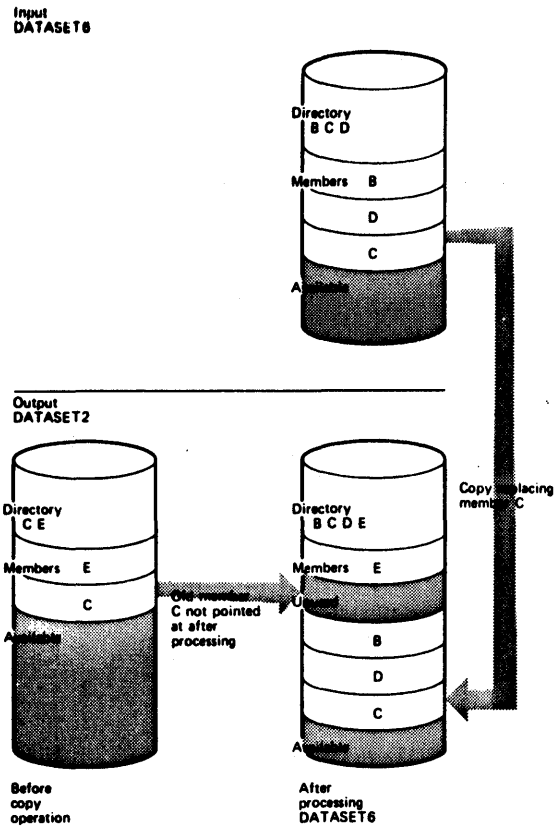


Figure 25. Copy Operation with "Replace" Specified on the Data Set Level

The example follows:

```
//COPY     JOB   ....
//JOBSTEP  EXEC  PGM=IEBCOPY
//SYSPRINT DD    SYSOUT=A
//INOUT2   DD    DSNAME=DATASET2,UNIT=3330-1,VOL=SER=111113,
//              DISP=OLD
//INOUT6   DD    DSNAME=DATASET6,UNIT=3350,VOL=SER=111117,
//              DISP=(OLD,KEEP)
//SYSUT3   DD    UNIT=SYSDA,SPACE=(TRK,(1))
//SYSUT4   DD    UNIT=SYSDA,SPACE=(TRK,(1))
//SYSIN    DD    *
COPYOPER  COPY  OUTDD=INOUT2
                INDD=((INOUT6,R))
/*
```

The control statements are discussed below:

- INOUT2 DD defines a partitioned data set (DATASET2), which resides on a 3330-1 volume. This data set contains two members (C and E).
- INOUT6 DD defines a partitioned data set (DATASET6), which resides on a 3350 volume. This data set contains three members (B, C, and D).
- SYSUT3 DD defines a temporary spill data set. One track is allocated on a disk volume.
- SYSUT4 DD defines a temporary spill data set. One track is allocated on a disk volume.
- SYSIN DD defines the control data set, which follows in the input stream. The data set contains a COPY statement and an INDD statement.
- COPY indicates the start of the copy operation. The absence of a SELECT or EXCLUDE statement causes a default to a full copy. The OUTDD parameter specifies INOUT2 as the DD statement for the output data set (DATASET2).
- INDD specifies INOUT6 as the DD statement for the input data set (DATASET6). Members B, C, and D are copied to the output data set (DATASET2). The pointer in the output data set directory is changed to point to the new (copied) member C; thus, the space occupied by the old member C is embedded unused space. Member C is copied even though the output data set already contains a member named "C" because the replace option is specified for all identically named members on the input data set; that is, the replace option is specified on the data set level.

The temporary spill data sets may or may not be opened, depending on the amount of virtual storage available; therefore, the SYSUT3 and SYSUT4 DD statements should always appear in the job stream.

IEBCOPY EXAMPLE 4

In this example, five members (A, C, D, E, and G) are selected from two input partitioned data sets (DATASET6 and DATASET2) copied to a new output partitioned data set (DATASET4). Figure 26 on page 68 shows the input and output data sets before and after processing.

```
//COPY     JOB     ....
//JOBSTEP  EXEC    PGM=IEBCOPY
//SYSPRINT DD      SYSOUT=A
//INOUT2   DD      DSNAME=DATASET2,UNIT=3330,VOL=SER=111114,
//                 DISP=(OLD,DELETE)
//INOUT6   DD      DSNAME=DATASET6,UNIT=3350,VOL=SER=111117,
//                 DISP=(OLD,KEEP)
//INOUT4   DD      DSNAME=DATASET4,UNIT=3350,VOL=SER=111116,
//                 DISP=(NEW,KEEP),SPACE=(TRK,(5,,2)),
//                 DCB=(RECFM=VB,LRECL=96,BLKSIZE=300)
//SYSUT3   DD      UNIT=SYSDA,SPACE=(TRK,(1))
//SYSUT4   DD      UNIT=SYSDA,SPACE=(TRK,(1))
//SYSIN    DD      *
COPYOPER  COPY     OUTDD=INOUT4
                   INDD=INOUT6
                   INDD=INOUT2
          SELECT   MEMBER=(C,D,E,A,G)
/*
```

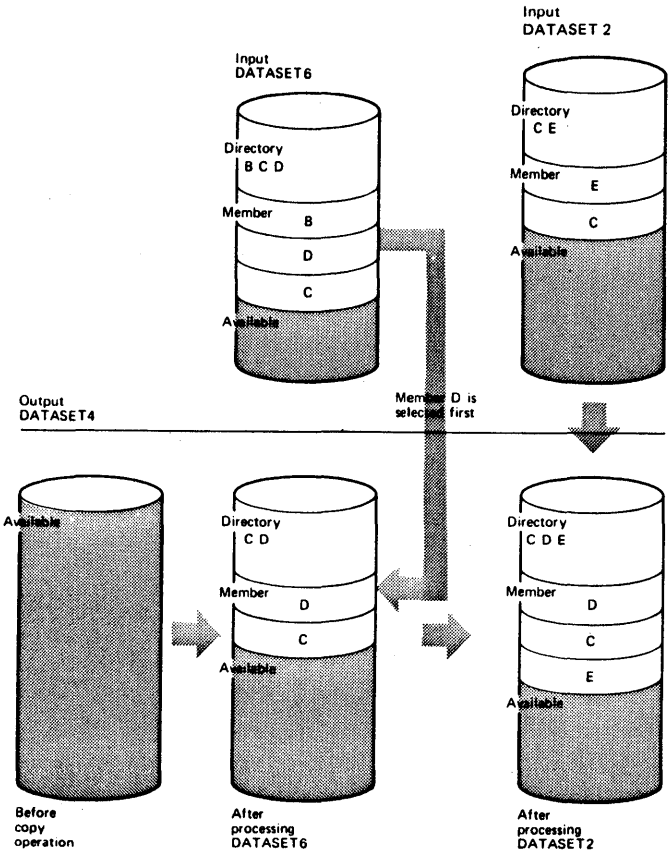


Figure 26. Copying Selected Members with Reblocking and Deblocking

The control statements are discussed below:

- INOUT2 DD defines a partitioned data set (DATASET2), which resides on a 3330 volume. This data set contains two members (C AND E) in variable-blocked format with a logical record length of 96 bytes and a block size of 500 bytes. This data set is to be deleted when processing is completed.
- INOUT6 DD defines a partitioned data set (DATASET6), which resides on a 3350 volume. This data set contains three members (B, C, and D) in variable-blocked format with a logical record length of 96 bytes and a block size of 100 bytes.
- INOUT4 DD defines a partitioned data set (DATASET4). This data set is new and is to be kept after the copy operation. Five tracks are allocated for the data set on a 3350 volume. Two blocks are allocated for directory entries. In addition, records are to be copied to this data set in variable blocked format with a logical record length of 96 bytes and a block size of 300 bytes.
- SYSUT3 DD defines a temporary spill data set. One track is allocated on a disk volume.

- SYSUT4 DD defines a temporary spill data set. One track is allocated on a disk volume.
- SYSIN DD defines the control data set, which follows in the input stream. The data set contains a COPY statement, two INDD statements, and a SELECT statement.
- COPY indicates the start of the copy operation. The use of a SELECT statement causes a selective copy. The OUTDD parameter specifies INOUT4 as the DD statement for the output data set (DATASET4).
- The first INDD statement specifies INOUT6 as the DD statement for the first input data set (DATASET6) to be processed. The members specified on the SELECT statement are searched for. The found members (C and D) are copied to the output data set (DATASET4) in the order in which they reside on the input data set, that is, in TTR (track record) order. In this case, member D is copied first, and then member C is copied.
- The second INDD statement specifies INOUT2 as the DD statement for the second input data set (DATASET2) to be processed. The members specified on the SELECT statement and not found on the first input data set are searched for. The found member (E) is copied onto the output data set (DATASET4). All members on DATASET2 are lost when the data set is deleted.
- SELECT specifies the members to be selected from the input data sets (DATASET6 and DATASET2) to be copied to the output data set (DATASET4).

The temporary spill data sets may or may not be opened, depending on the amount of virtual storage available; therefore, the SYSUT3 and SYSUT4 DD statements should always appear in the job stream.

IEBCOPY EXAMPLE 5

In this example, two members (A and B) are selected from two input partitioned data sets (DATASET5 and DATASET6) copied to an existing output partitioned data set (DATASET1). Member B replaces an Identically named member that already exists on the output data set. Figure 27 on page 70 shows the input and output data sets before and after processing.

```
//COPY     JOB      ....
//JOBSTEP  EXEC     PGM=IEBCOPY
//SYSPRINT DD       SYSOUT=A
//INOUT1   DD       DSNAME=DATASET1,UNIT=3330,VOL=SER=111112,
//                  DISP=(OLD,KEEP)
//INOUT6   DD       DSNAME=DATASET6,UNIT=3350,VOL=SER=111115,
//                  DISP=OLD
//INOUT5   DD       DSNAME=DATASET5,UNIT=3330,VOL=SER=111116,
//                  DISP=(OLD,KEEP)
//SYSUT3   DD       UNIT=SYSDA,SPACE=(TRK,(1))
//SYSUT4   DD       UNIT=SYSDA,SPACE=(TRK,(1))
//SYSIN    DD       *
COPYOPER   COPY     OUTDD=INOUT1
                    INDD=INOUT5,INOUT6
           SELECT   MEMBER=((B,,R),A)
/*
```

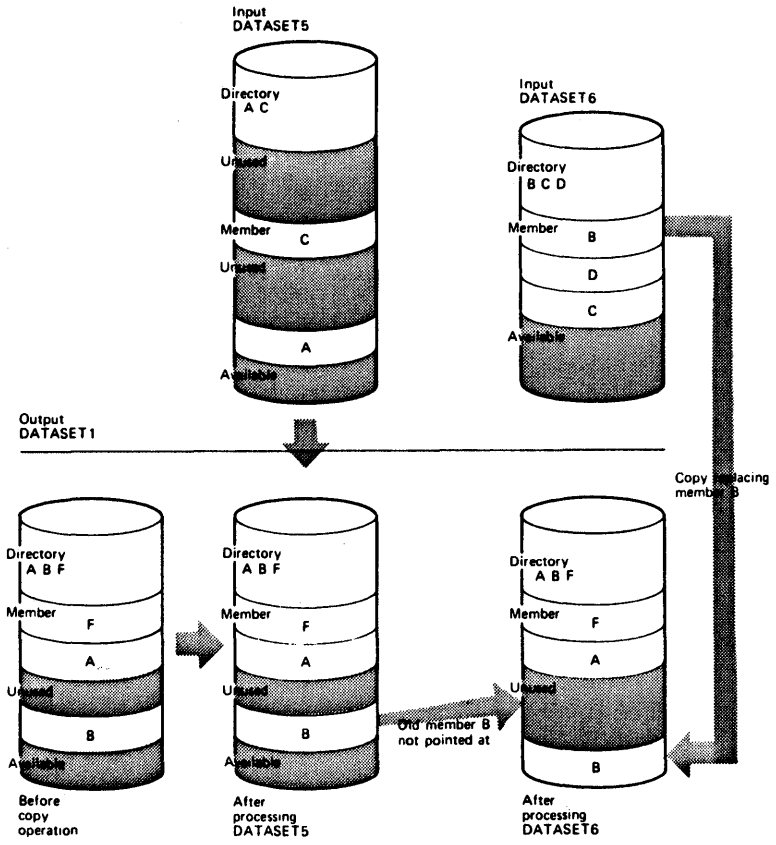


Figure 27. Selective Copy with "Replace" Specified on the Member Level

The control statements are discussed below:

- INOUT1 DD defines a partitioned data set (DATASET1). This data set resides on a 3330 volume and contains three members (A, B, and F).
- INOUT6 DD defines a partitioned data set (DATASET6). This data set resides on a 3350 volume and contains three members (B, C, and D).
- INOUT5 DD defines a partitioned data set (DATASET5). This data set resides on a 3330 volume and contains two members (A and C).
- SYSUT3 DD defines a temporary spill data set. One track is allocated on a disk volume.
- SYSUT4 DD defines a temporary spill data set. One track is allocated on a disk volume.
- SYSIN DD defines the control data set, which follows in the input stream. The data set contains a COPY statement, an INDD statement, and a SELECT statement.
- COPY indicates the start of the copy operation. The use of a SELECT statement causes a selective copy. The OUTDD parameter specifies INOUT1 as the DD statement for the output data set (DATASET1).
- INDD specifies INOUT5 as the DD statement for the first input data set (DATASET5) to be processed and INOUT6 as the DD statement for the second input data set (DATASET6) to be processed. Processing occurs, as follows: (1) selected members are searched for on DATASET5, (2) member A is found, but is not copied to the output data set because it already exists on DATASET2 and the replace option is not specified, (3) selected members not found on DATASET5 are searched for on DATASET6, and (4) member B is found and copied to the output data set (DATASET1), even though a member named B already exists on the output data set, because the replace option is specified for member B on the member level. The pointer in the output data set directory is changed to point to the new (copied) member B; thus, the space occupied by the old member B is unused.
- SELECT specifies the members to be selected from the input data sets (DATASET5 and DATASET6) to be copied to the output data set (DATASET1).

The temporary spill data sets may or may not be opened, depending on the amount of virtual storage available; therefore, the SYSUT3 and SYSUT4 DD statements should always appear in the job stream.

IEBCOPY EXAMPLE 6

In this example, two members (A and B) are selected from two input partitioned data sets (DATASET5 and DATASET6) copied to an existing output partitioned data set (DATASET1). All members found on DATASET5 replace identically named members on DATASET1. Figure 28 on page 72 shows the input and output data sets before and after processing.

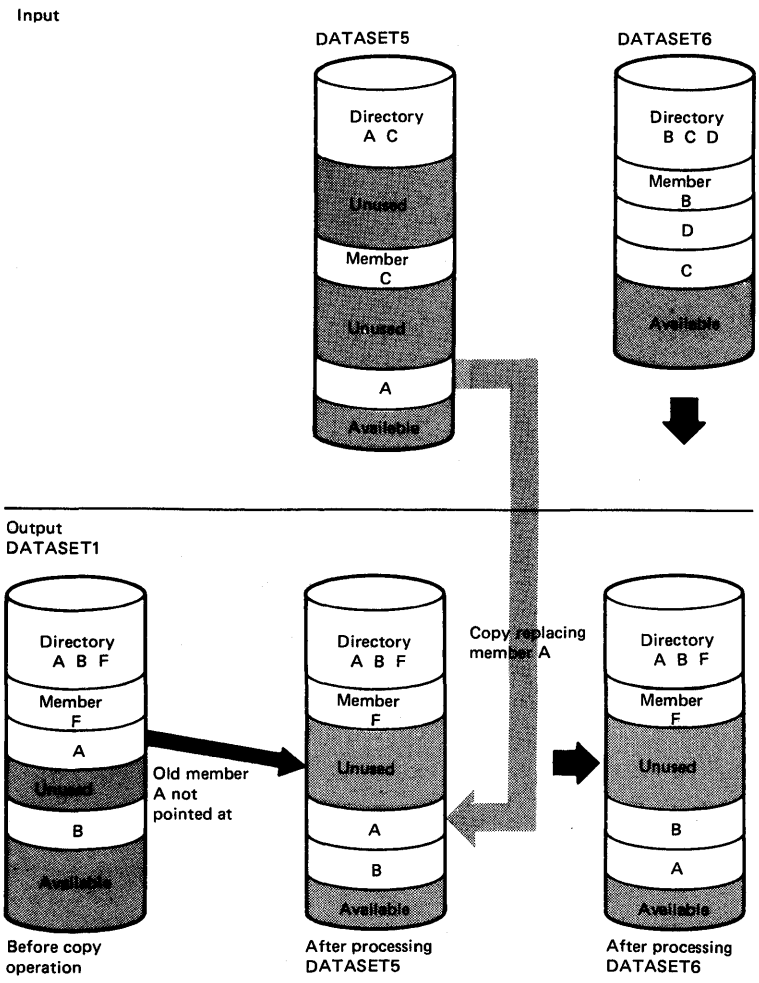


Figure 28. Selective Copy with "Replace" Specified on the Data Set Level

```
//COPY     JOB      ....
//JOBSTEP  EXEC     PGM=IEBCOPY
//SYSPRINT DD       SYSOUT=A
//INOUT1   DD       DSNAME=DATASET1,UNIT=3350,VOL=SER=111112,
//                  DISP=(OLD,KEEP)
//INOUT5   DD       DSNAME=DATASET5,UNIT=3330,VOL=SER=111114,
//                  DISP=(OLD,DELETE)
//INOUT6   DD       DSNAME=DATASET6,UNIT=2305-2,VOL=SER=111115
//                  DISP=(OLD,KEEP)
//SYSUT3   DD       UNIT=SYSDA,SPACE=(TRK,(1))
//SYSUT4   DD       UNIT=SYSDA,SPACE=(TRK,(1))
//SYSIN    DD       *
COPYOPER   COPY     OUTDD=INOUT1
                    INDD=((INOUT5,R),INOUT6)
           SELECT   MEMBER=(A,B)
/*
```

The control statements are discussed below:

- INOUT1 DD defines a partitioned data set (DATASET1). This data set resides on a 3350 volume and contains three members (A, B, and F).
- INOUT5 DD defines a partitioned data set (DATASET5). This data set contains two members (A and C) and resides on a 3330 volume. This data set is to be deleted when processing is completed.
- INOUT6 DD defines a partitioned data set (DATASET6). This data set contains three members (B, C, and D) and resides on a 2305-2 volume.
- SYSUT3 DD defines a temporary spill data set. One track is allocated on a disk volume.
- SYSUT4 DD defines a temporary spill data set. One track is allocated on a disk volume.
- SYSIN DD defines the control data set, which follows in the input stream. The data set contains a COPY statement, an INDD statement, and a SELECT statement.
- COPY indicates the start of the copy operation. The presence of a SELECT statement causes a selective copy. The OUTDD operand specifies INOUT1 as the DD statement for the output data set (DATASET1).
- INDD specifies INOUT5 as the DD statement for the first input data set (DATASET5) to be processed and INOUT6 as the statement for the second input data set (DATASET6) to be processed. Processing occurs, as follows: (1) selected members are searched for on DATASET5, (2) member A is found and copied to the output data set (DATASET1) because the replace option was specified on the data set level for DATASET5, (3) member B, which was not found on DATASET5 is searched for and found on DATASET6, (4) member B is not copied because DATASET1 already contains a member called member B and the replace option is not specified for DATASET6. The pointer in the output data set directory is changed to point to the new (copied) member A; thus, the space occupied by the old member A is unused.
- SELECT specifies the members to be selected from the input data sets (DATASET5 and DATASET6) to be copied to the output data set (DATASET1).

The temporary spill data sets may or may not be opened, depending on the amount of virtual storage available; therefore, the SYSUT3 and SYSUT4 DD statements should always appear in the job stream.

IEBCOPY EXAMPLE 7

In this example, four members (A, B, C, and D) are selected from an input partitioned data set (DATASET6) copied to an existing output partitioned data set (DATASET3). Member B is renamed H; member C is renamed J; and member D is renamed K. In addition, member C (renamed J) replaces the identically named member (J) on the output partitioned data set. Figure 29 shows the input and output data sets before and after processing.

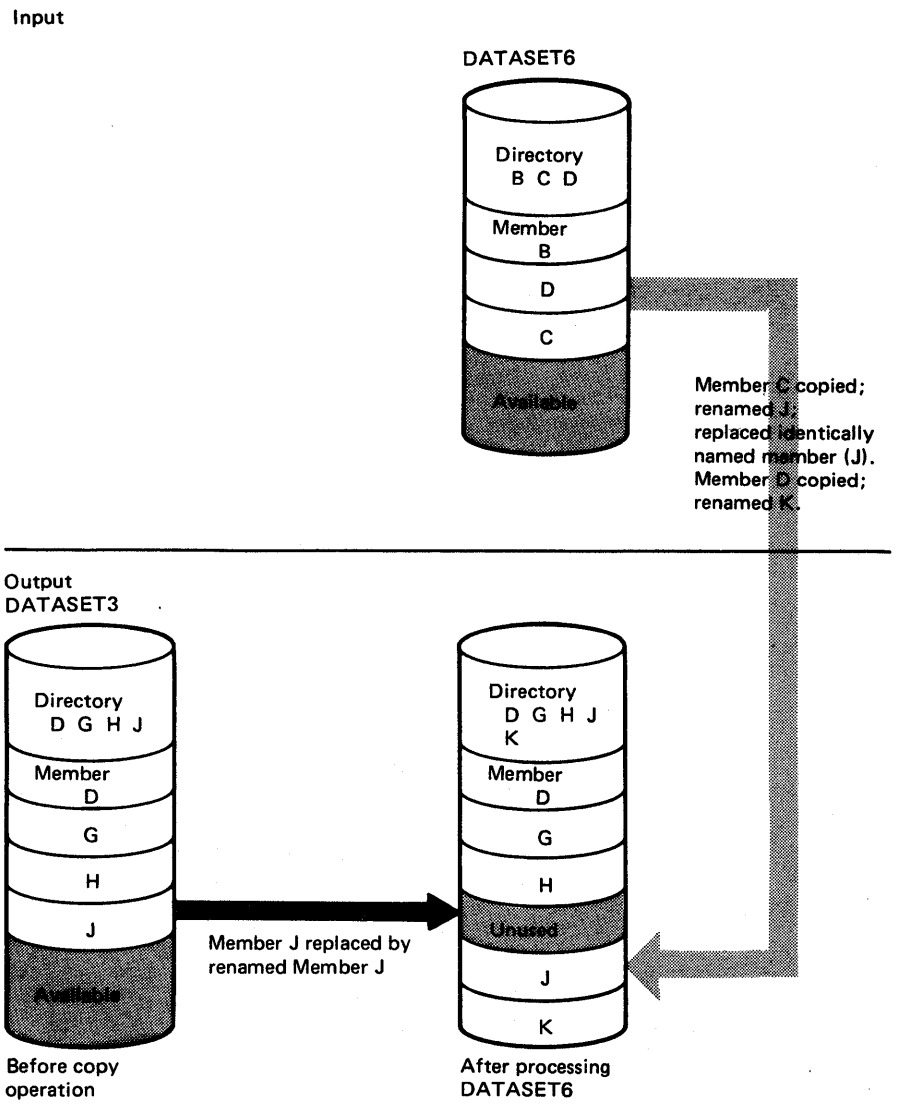


Figure 29. Renaming Selected Members Using IEBCOPY

```
//COPY     JOB      ....
//JOBSTEP  EXEC     PGM=IEBCOPY
//SYSPRINT DD       SYSOUT=A
//INOUT3   DD       DSNAME=DATASET3,UNIT=disk,VOL=SER=111114,
//                  DISP=(OLD,KEEP)
//INOUT6   DD       DSNAME=DATASET6,UNIT=disk,VOL=SER=111117,
//                  DISP=(OLD,DELETE)
//SYSUT3   DD       UNIT=SYSDA,SPACE=(TRK,(1))
//SYSUT4   DD       UNIT=SYSDA,SPACE=(TRK,(1))
//SYSIN    DD       *
COPYOPER   COPY     OUTDD=INOUT3,INDD=INOUT6
           SELECT   MEMBER=((B,H),(C,J,R),A,(D,K))
/*
```

The control statements are discussed below:

- INOUT3 DD defines a partitioned data set (DATASET3). This data set contains four members (D, G, H, and J) and resides on a disk volume.
- INOUT6 DD defines a partitioned data set (DATASET6). This data set contains three members (B, C, and D) and resides on a disk volume. DATASET6 is to be deleted when processing is completed; thus, all members on this data set are lost.
- SYSUT3 DD defines a temporary spill data set. One track is allocated on a disk volume.
- SYSUT4 DD defines a temporary spill data set. One track is allocated on a disk volume.
- SYSIN DD defines the control data set, which follows in the input stream. The data set contains a COPY statement, an INDD statement, and a SELECT statement.
- COPY indicates the start of the copy operation. The presence of a SELECT statement causes a selective copy. The OUTDD parameter specifies INOUT3 as the DD statement for the output data set (DATASET3).
- INDD specifies INOUT6 as the DD statement for the input data set (DATASET6). Processing occurs, as follows:

  1. Selected members are searched for on DATASET6.
  2. Member B is found, but is not copied to DATASET3 because its intended new name (H) is identical to the name of a member (H), which already exists on the output data set, and replace is not specified.
  3. Member C is found and copied to the output data set (DATASET3), although its new name (J) is identical to the name of a member (J), which already exists on the output data set, because the replace option is specified for the renamed member.
  4. Member D is copied onto the output data set (DATASET3) because its new name (K) does not already exist there.

- SELECT specifies the members to be selected from the input data set (DATASET6) to be copied to the output data set (DATASET3).

  The temporary spill data sets may or may not be opened, depending on the amount of virtual storage available; therefore, the SYSUT3 and SYSUT4 DD statements should always appear in the job stream.

IEBCOPY EXAMPLE 8

In this example, five members (A, B, C, J, and L) are excluded from the copy operation when each of the input partitioned data sets (DATASET1, DATASET3, and DATASET6) is processed. In addition, replace is specified for the last input partitioned data set (DATASET6) to be processed; thus, with the exception of the members specified on the EXCLUDE statement, all members on DATASET6 will replace any identically named members on the output partitioned data set (DATASET4). Figure 30 on page 77 shows the input and output data sets before and after processing.

```
//COPY     JOB      ....
//JOBSTEP  EXEC     PGM=IEBCOPY
//SYSPRINT DD       SYSOUT=A
//INOUT1   DD       DSNAME=DATASET1,UNIT=disk,VOL=SER=111112,
//                  DISP=(OLD,KEEP)
//INOUT3   DD       DSNAME=DATASET3,UNIT=disk,VOL=SER=111114,
//                  DISP=OLD
//INOUT4   DD       DSNAME=DATASET4,UNIT=disk,VOL=SER=111115,
//                  DISP=(NEW,KEEP),SPACE=(TRK,(3,1,2)),
//                  DCB=(LRECL=100,RECFM=FB,BLKSIZE=400)
//INOUT6   DD       DSNAME=DATASET6,UNIT=disk,VOL=SER=111116,
//                  DISP=OLD
//SYSUT3   DD       UNIT=SYSDA,SPACE=(TRK,(1))
//SYSUT4   DD       UNIT=SYSDA,SPACE=(TRK,(1))
//SYSIN    DD       *
COPYOPER COPY       OUTDD=INOUT4,
                    INDD=INOUT1,INOUT3,(INOUT6,R)
         EXCLUDE    MEMBER=(A,J,B,L,C)
/*
```

The control statements are discussed below:

- INOUT1 DD defines a partitioned data set (DATASET1). This data set contains three members (A, B, and F) and resides on a disk volume. The record format is fixed-blocked with a logical record length of 100 bytes and a block size of 400 bytes.
- INOUT3 DD defines a partitioned data set (DATASET3), which resides on a disk volume. This data set contains four members (D, G, H, and J) in fixed-blocked format with a logical record length of 100 bytes and a block size of 600 bytes.
- INOUT4 DD defines a new partitioned data set (DATASET4). Three tracks are allocated for the copied members on a disk volume. Two blocks are allocated for directory entries. In addition, records are to be copied to this data set in fixed-blocked format with a logical record length of 100 bytes and a block size of 400 bytes.
- INOUT6 DD defines a partitioned data set (DATASET6). This data set contains three members (B, C, and D) in fixed format. The records have a logical record length of 100 bytes and a block size of 100 bytes. This data set resides on a disk volume.
- SYSUT3 DD defines a temporary spill data set. One track is allocated on a disk volume.
- SYSUT4 DD defines a temporary spill data set. One track is allocated on a disk volume.
- SYSIN DD defines the control data set, which follows in the input stream. The data set contains a COPY statement and an EXCLUDE statement.

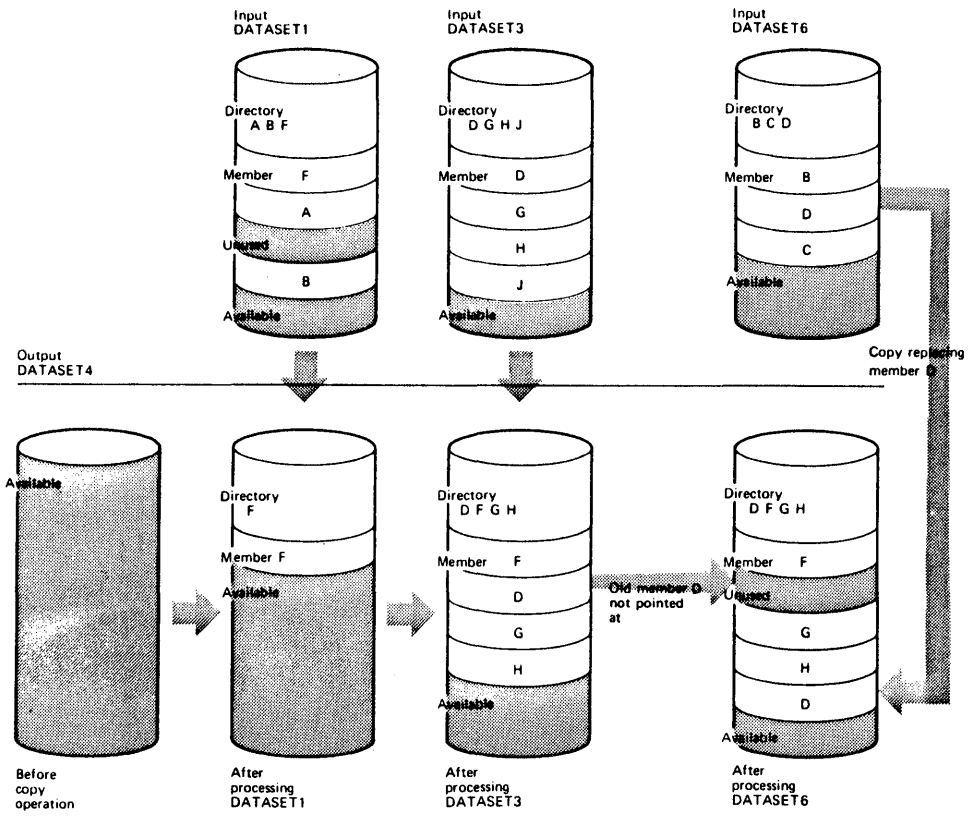


Figure 30. Exclusive Copy with "Replace" Specified for One Input Partitioned Data Set

- COPY indicates the start of the copy operation. The presence of an EXCLUDE statement causes an exclusive copy. The OUTDD parameter specifies INOUT4 as the DD statement for the output data set (DATASET4). The INDD parameter specifies INOUT1 as the DD statement for the first input data set (DATASET1) to be processed, INOUT3 as the DD statement for the second input data set (DATASET3) to be processed, and INOUT6 as the DD statement for the last input data set (DATASET6) to be processed. Processing occurs, as follows:

  1. Member F, which is not named on the EXCLUDE statement, is copied from DATASET1.

  2. Members D, G, and H, which are not named on the EXCLUDE statement, are copied from DATASET3.

  3. Member D is copied from DATASET6 because the replace option is specified for nonexcluded members.

  The pointer in the output data set directory is changed to point at the new (copied) member D; thus, the space occupied by the **old** member D (copied from DATASET3) is unused.

- EXCLUDE specifies the members to be excluded from the copy operation. The named members are excluded from all of the input partitioned data sets specified in the copy operation.

The temporary spill data sets may or may not be opened, depending on the amount of virtual storage available; therefore, the SYSUT3 and SYSUT4 DD statements should always appear in the job stream.

IEBCOPY EXAMPLE 9

In this example, a partitioned data set is unloaded to a tape volume to create a backup copy of the data set. If this step is successful, the partitioned data set is to be compressed in place.

```
//SAVE       JOB  ....
//STEP1      EXEC PGM=IEBCOPY
//SYSPRINT   DD   SYSOUT=A
//INPDS      DD   DSNAME=PARTPDS,UNIT=disk,VOL=SER=PCP001,
//               DISP=OLD
//BACKUP     DD   DSNAME=SAVDATA,UNIT=tape,VOL=SER=TAPE03,
//               DISP=(NEW,KEEP),LABEL=(,SL)
//SYSUT3     DD   DSNAME=TEMP1,UNIT=disk,VOL=SER=111111,
//               DISP=(NEW,DELETE),SPACE=(80,(60,45))
//SYSIN      DD   *
           COPY OUTDD=BACKUP,INDD=INPDS
/*
//STEP2      EXEC PGM=IEBCOPY,COND=(0,NE),
//               PARM='SIZE=99999999K'
//SYSPRINT   DD   SYSOUT=A
//COMPDS     DD   DSNAME=PARTPDS,UNIT=disk,DISP=OLD,
//               VOL=SER=PCP001
//SYSUT3     DD   DSNAME=TEMPA,UNIT=disk,VOL=SER=111111,
//               DISP=(NEW,DELETE),SPACE=(80,(60,45))
//SYSUT4     DD   DSNAME=TEMPB,UNIT=disk,VOL=SER=111111,
//               SPACE=(256,(15,1)),DCB=KEYLEN=8
//SYSIN      DD   *
           COPY OUTDD=COMPDS,INDD=COMPDS
/*
```

The control statements are discussed below:

- INPDS DD defines a partitioned data set (PARTPDS) that resides on a disk volume and is assumed to have 700 members. The number of members is used to calculate the space allocation on SYSUT3.
- BACKUP DD defines a sequential data set to hold PARTPDS in unloaded form. Block size information can optionally be added; this data set must be NEW.
- SYSUT3 DD defines the temporary spill data set.
- SYSIN DD defines the control data set, which follows in the input stream. The data set contains a COPY statement.
- COPY marks the beginning of the unload operation; the absence of an EXCLUDE or SELECT statement causes the entire partitioned data set (INDD=INPDS) to be unloaded to a sequential data set (OUTDD=BACKUP).
- The second EXEC statement marks the beginning of the compress-in-place operation. The SIZE parameter indicates that the buffers are to be as large as possible. The COND parameter indicates that the compress-in-place is to be performed only if the unload operation was successful.
- COMPDS DD defines a partitioned data set (PARTPDS) that contains 700 members and resides on a disk volume.
- SYSUT3 DD defines the temporary spill data set to be used if there is not enough space in main storage for the input data set's directory entries. TEMPA contains one 80-character record for each member.
- SYSUT4 DD defines the temporary spill data set to be used if there is not enough space in main storage for the output

partitioned data set's directory blocks. TEMPB contains one 256-character record for each directory block.

- SYSIN DD defines the control data set, which follows in the input stream. The data set contains a COPY statement.
- COPY marks the beginning of the copy operation. The absence of a SELECT or EXCLUDE statement causes a default to a full copy. Because the same DD statement is specified for both the INDD and OUTDD operands, the data set is compressed in place.

The temporary spill data sets may or may not be opened, depending on the amount of virtual storage available; therefore, the SYSUT3 and SYSUT4 DD statements should always appear in the job stream. However, the SYSUT4 data set is never used for an unload operation.

For an unload operation, only one INDD data set may be specified for one OUTDD data set.

IEBCOPY EXAMPLE 10

In this example, two input partitioned data sets (DATASET5 and DATASET6) are copied to an existing output partitioned data set (DATASET1). In addition, all members on DATASET6 are copied; members on the output data set that have the same names as the copied members are replaced. After DATASET6 is processed, the output data set (DATASET1) is compressed in place. Figure 31 on page 80 shows the input and output data sets before and after processing.

```
//COPY     JOB   ....
//JOBSTEP  EXEC  PGM=IEBCOPY
//SYSPRINT DD    SYSOUT=A
//INOUT1   DD    DSNAME=DATASET1,UNIT=3330,VOL=SER=111112,
//              DISP=(OLD,KEEP)
//INOUT5   DD    DSNAME=DATASET5,UNIT=3350,VOL=SER=111114,
//              DISP=OLD
//INOUT6   DD    DSNAME=DATASET6,UNIT=3350,VOL=SER=111115,
//              DISP=(OLD,KEEP)
//SYSUT3   DD    UNIT=SYSDA,SPACE=(TRK,(1))
//SYSUT4   DD    UNIT=SYSDA,SPACE=(TRK,(1))
//SYSIN    DD    *
COPYOPER  COPY   OUTDD=INOUT1
                 INDD=INOUT5,(INOUT6,R),INOUT1
/*
```

The control statements are discussed below:

- INOUT1 DD defines a partitioned data set (DATASET1). This data set contains three members (A, B, and F) and resides on a 3330 volume.
- INOUT5 DD defines a partitioned data set (DATASET5). This data set contains two members (A and C) and resides on a 3350 volume.
- INOUT6 DD defines a partitioned data set (DATASET6). This data set contains three members (B, C, and D) and resides on a 3350 volume.
- SYSUT3 DD defines a temporary spill data set. One track is allocated on a disk volume.
- SYSUT4 DD defines a temporary spill data set. One track is allocated on a disk volume.

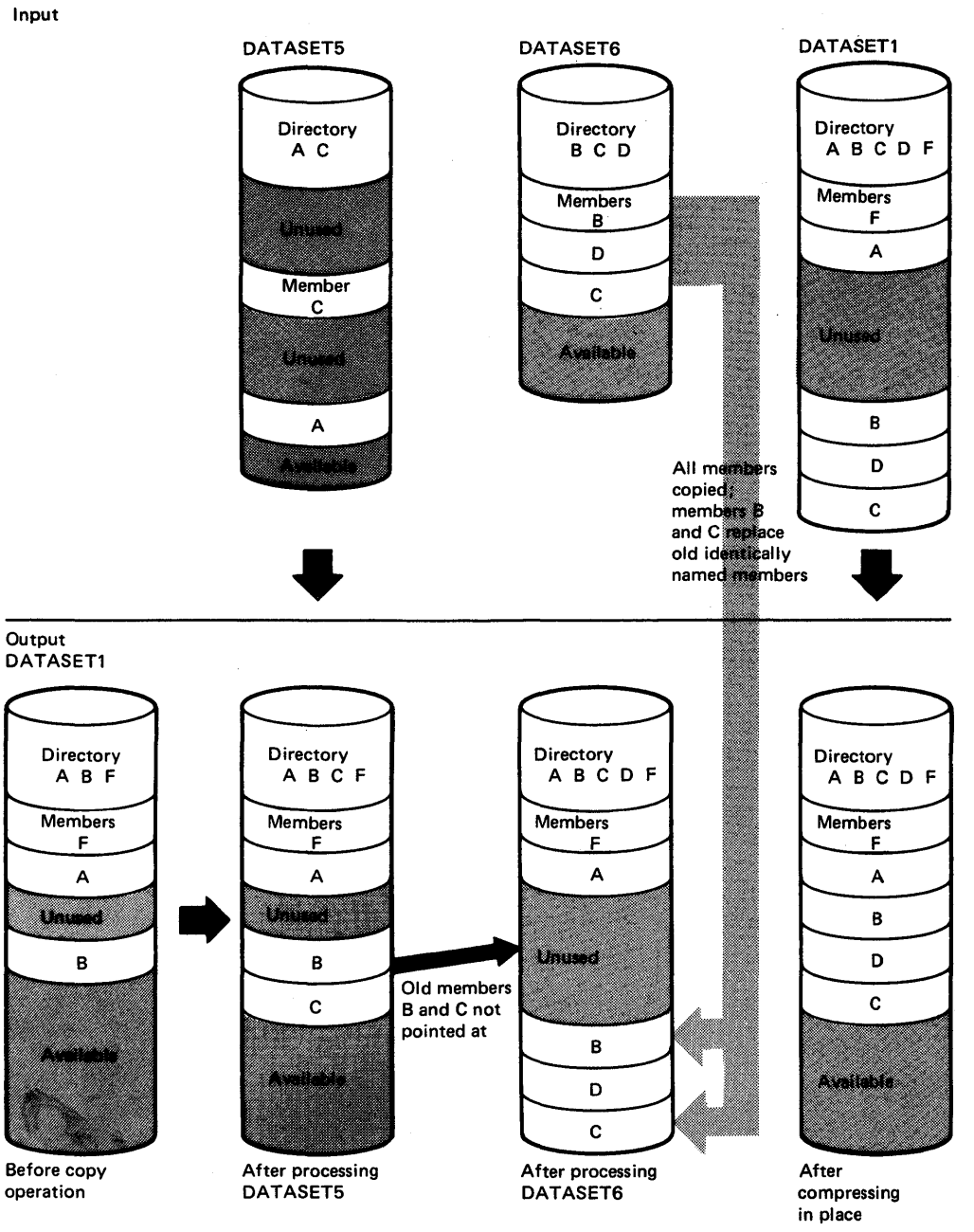


Figure 31. Compress-in-Place Following Full Copy with "Replace" Specified

- SYSIN DD defines the control data set, which follows in the input stream. The data set contains a COPY statement and an INDD statement.
- COPY indicates the start of the copy operation. The OUTDD operand specifies INOUT1 as the DD statement for the output data set (DATASET1). The absence of a SELECT or EXCLUDE statement causes a default to a full copy.
- INDD specifies INOUT5 as the DD statement for the first input data set (DATASET5) to be processed. It then specifies INOUT6 as the DD statement for the second input data set (DATASET6) to be processed; in addition, the replace option is specified for all members copied from DATASET6. Finally, it specifies INOUT1 as the DD statement for the last input data set (DATASET1) to be processed; this causes a compress-in-place of DATASET1 because it is also specified as the output data set. Processing occurs, as follows:

  1. Member A is not copied from DATASET5 onto the output data set (DATASET1) because it already exists on DATASET1 and the replace option was not specified for DATASET5.
  2. Member C is copied from DATASET5 to the output data set (DATASET1), occupying the first available space.
  3. All members are copied from DATASET6 to the output data set (DATASET1), immediately following the last member. Members B and C are copied even though the output data set already contains members with the same names because the replace option is specified on the data set level.

  The pointers in the output data set directory are changed to point to the new members B and C; thus, the space occupied by the old members B and C is unused. The members currently on DATASET1 are compressed in place, thereby eliminating embedded unused space.

The temporary spill data sets may or may not be opened, depending on the amount of virtual storage available; therefore, the SYSUT3 and SYSUT4 DD statements should always appear in the job stream.

IEBCOPY EXAMPLE 11

In this example, members are selected, excluded, and copied from input partitioned data sets onto an output partitioned data set. This example is designed to illustrate multiple copy operations. Figure 32 on page 82 shows the input and output data sets before and after processing.

Compress-in-Place Operation

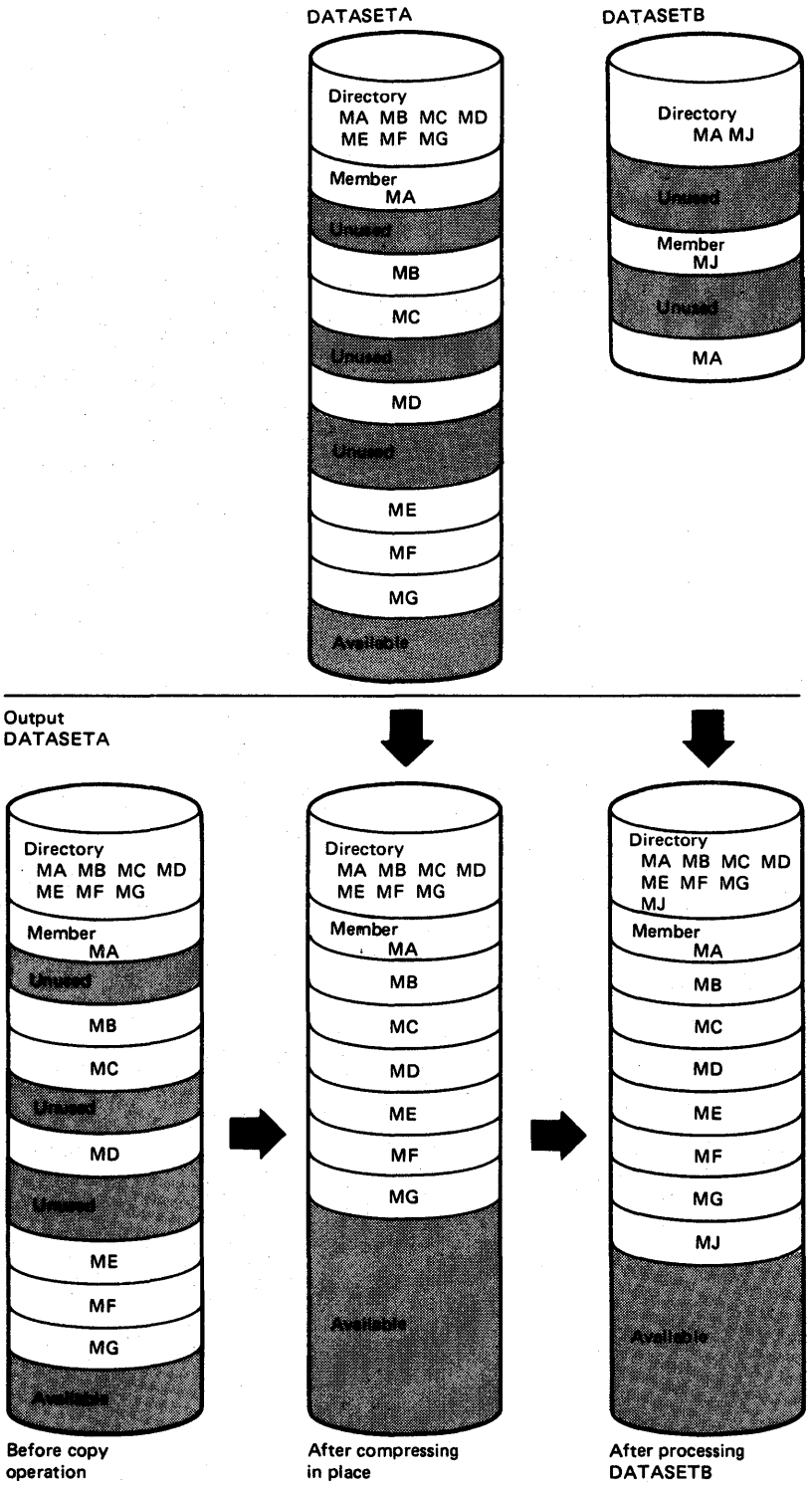


Figure 32 (Part 1 of 2). Multiple Copy Operations/Copy Steps

Multiple Copy Steps

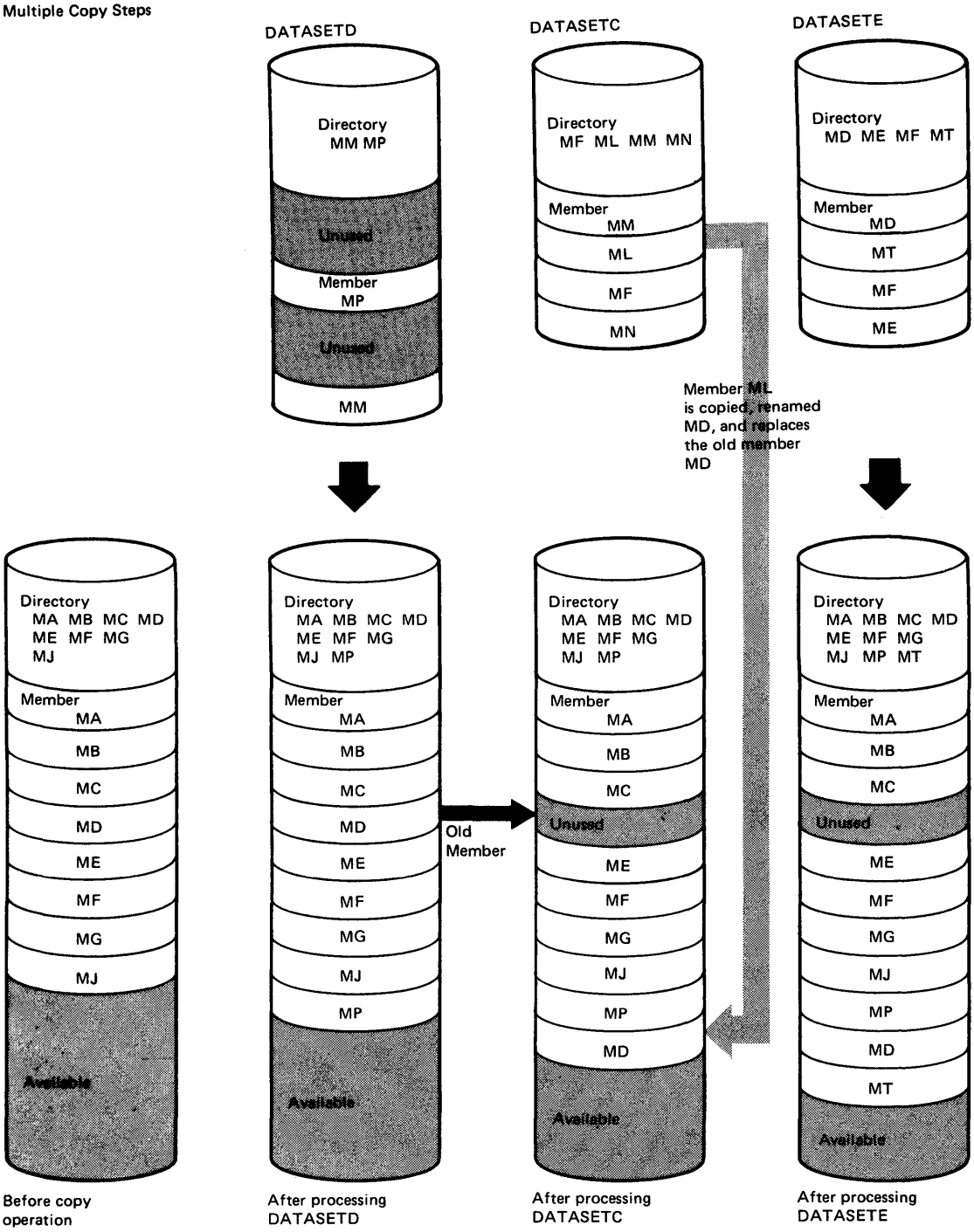


Figure 32 (Part 2 of 2). Multiple Copy Operations/Copy Steps

```
//COPY     JOB       ....
//JOBSTEP  EXEC      PGM=IEBCOPY
//SYSPRINT DD        SYSOUT=A
//INOUTA   DD        DSNAME=DATASETA,UNIT=disk,VOL=SER=111113,
//                   DISP=OLD
//INOUTB   DD        DSNAME=DATASETB,UNIT=disk,VOL=SER=111115,
//                   DISP=(OLD,KEEP)
//INOUTC   DD        DSNAME=DATASETC,UNIT=disk,VOL=SER=111114,
//                   DISP=(OLD,KEEP)
//INOUTD   DD        DSNAME=DATASETD,UNIT=disk,VOL=SER=111116,
//                   DISP=OLD
//INOUTE   DD        DSNAME=DATASETE,UNIT=disk,VOL=SER=111117,
//                   DISP=OLD
//INOUTX   DD        DSNAME=DATASETX,UNIT=disk,VOL=SER=111112,
//                   DISP=(NEW,KEEP),SPACE=(TRK,(3,1,2))
//SYSUT3   DD        UNIT=SYSDA,SPACE=(TRK,(1))
//SYSUT4   DD        UNIT=SYSDA,SPACE=(TRK,(1))
//SYSIN    DD        *
COPERST1   COPY      O=INOUTX,I=INOUTA
           COPY      OUTDD=INOUTA,INDD=INOUTA
                     INDD=INOUTB
           COPY      O=INOUTA
                     INDD=INOUTD
           EXCLUDE   MEMBER=MM
                     INDD=INOUTC
           SELECT    MEMBER=((ML,MD,R))
                     INDD=INOUTE
/*
```

The control statements are discussed below:

- INOUTA DD defines a partitioned data (DATASETA). This data set contains seven members (MA, MB, MC, MD, ME, MF, and MG) and resides on a disk volume.
- INOUTB DD defines a partitioned data set (DATASETB). This data set resides on a disk volume and contains two members (MA and MJ).
- INOUTC DD defines a partitioned data set (DATASETC), that resides on a disk volume. The data set contains four members (MF, ML, MM, and MN).
- INOUTD DD defines a partitioned data set (DATASETD). This data set resides on a disk volume and contains two members (MM and MP).
- INOUTE DD defines a partitioned data set (DATASETE). This data set contains four members (MD, ME, MF, and MT) and resides on a disk volume.
- INOUTX DD defines a partitioned data set (DATASETX). This data set is new and is to be kept after the copy operation. Three tracks are allocated for the data set on a disk volume. Two blocks are allocated for directory entries.
- SYSUT3 DD defines a temporary spill data set. One track is allocated on a disk volume.
- SYSUT4 DD defines a temporary spill data set. One track is allocated on a disk volume.
- SYSIN DD defines the control data set, which follows in the input stream. The data set contains two COPY statements, several INDD statements, a SELECT statement, and an EXCLUDE statement.
- The first COPY statement indicates the start of the first copy operation. This copy operation is done to create a

backup copy of DATASETA, which is compressed in place in the second copy operation.

- The second COPY statement indicates the start of another copy operation. The absence of a SELECT or EXCLUDE statement causes a default to a full copy; however, the same DD statement, INOUTA, is specified for both the INDD and OUTDD parameters, causing a compress-in-place of the specified data set.

  The output data set is compressed in place first to save space because it is known that it contains embedded, unused space.

  INDD specifies INOUTB as the DD statement for the input data set (DATASETB) to be copied. Only member MJ is copied because member MA already exists on the output data set.

- The third COPY statement indicates the start of the third copy operation. The OUTDD parameter specifies INOUTA as the DD statement for the output data set (DATASETA). This copy operation contains more than one copy step.

  The first INDD statement specifies INOUTD as the DD statement for the first input data set (DATASETD) to be processed. Only member MP is copied to the output data set (DATASETA) because member MM is specified on the EXCLUDE statement. EXCLUDE specifies the member to be excluded from the first copy step within this copy operation.

  The second INDD statement marks the beginning of the second copy step for this copy operation and specifies INOUTC as the DD statement for the second input data set (DATASETC) to be processed. Member ML is searched for, found, and copied to the output data set (DATASETA). Member ML is copied even though its new name (MD) is identical to the name of a member (MD) that already exists on the output data set, because the replace option is specified for the renamed member.

  SELECT specifies the member to be selected from the input data set (DATASETC) to be copied to the output partitioned data set.

  The third INDD statement marks the beginning of the third copy step for this copy operation and specifies INOUTE as the DD statement for the last data set (DATASETE) to be copied. Only member MT is copied because the other members already exist on the output data set. Because the INDD statement is not followed by an EXCLUDE or SELECT statement, a full copy is performed.

The temporary spill data sets may or may not be opened, depending on the amount of virtual storage available; therefore, it is suggested that the SYSUT3 and SYSUT4 DD statements always appear in the job stream.

The output data set is compressed in place first to save space because it is known that it contains embedded, unused space.

IEBCOPY EXAMPLE 12

In this example, members are selected, excluded, and copied from input partitioned data sets to an output partitioned data set. This example is designed to illustrate multiple copy operations. Figure 33 on page 86 shows the input and output data sets before and after processing.

First copy operation

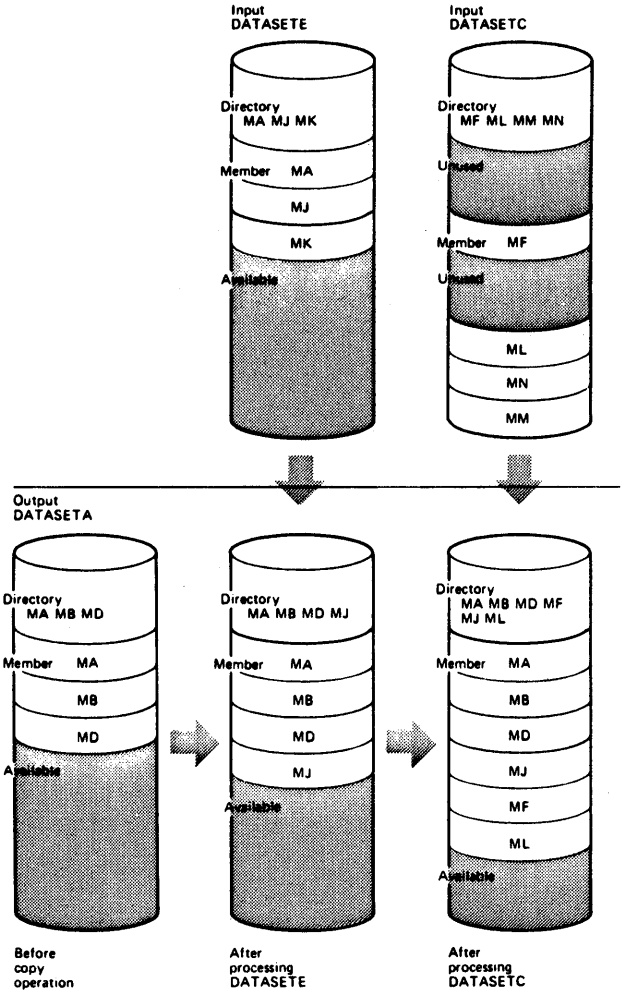


Figure 33 (Part 1 of 3). Multiple Copy Operations/Copy Steps within a Job Step

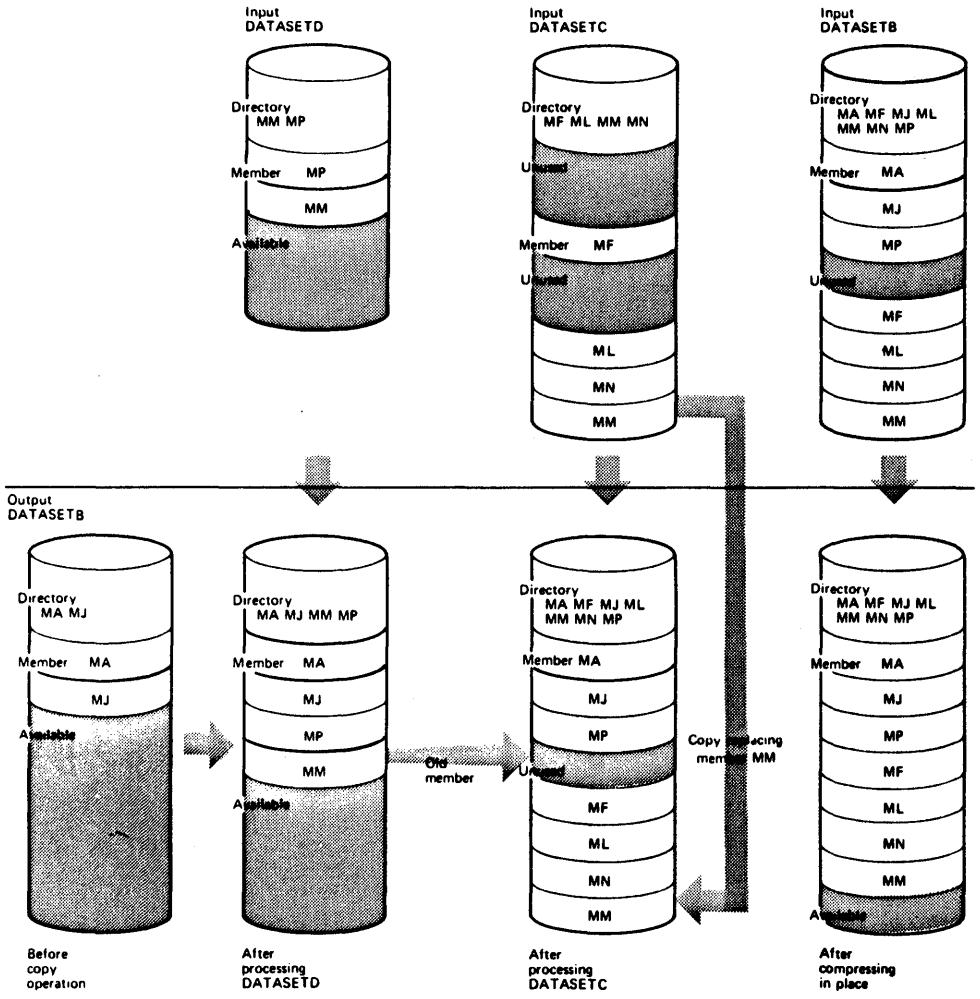


Figure 33 (Part 2 of 3). Multiple Copy Operations/Copy Steps within a Job Step

Third copy operation

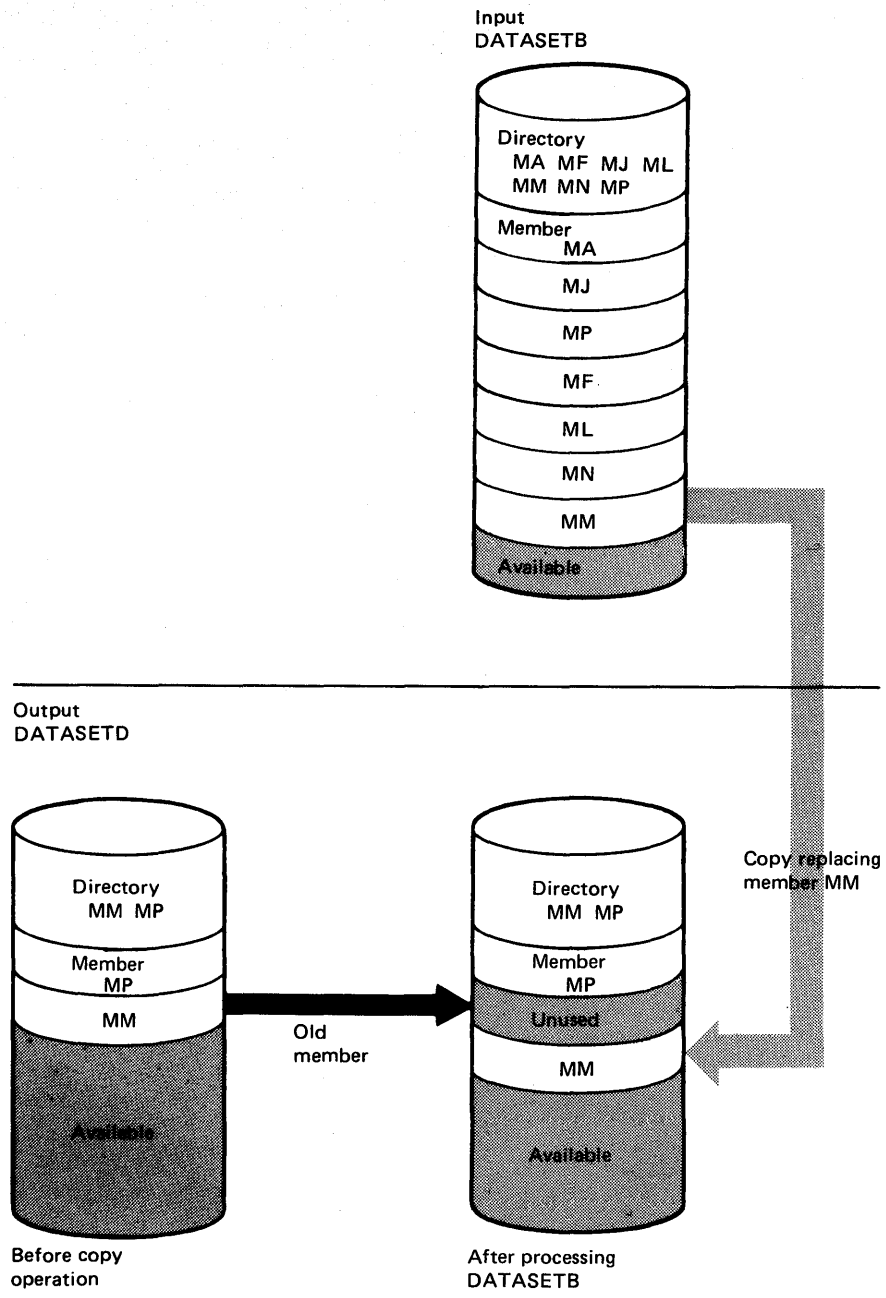


Figure 33 (Part 3 of 3). Multiple Copy Operations/Copy Steps within a Job Step

```
//COPY     JOB       ....
//JOBSTEP  EXEC      PGM=IEBCOPY
//SYSPRINT DD        SYSOUT=A
//INOUTA   DD        DSNAME=DATASETA,UNIT=disk,VOL=SER=111113,
//                   DISP=OLD
//INOUTB   DD        DSNAME=DATASETB,VOL=SER=111115,UNIT=disk,
//                   DISP=(OLD,KEEP)
//INOUTC   DD        DSNAME=DATASETC,VOL=SER=111114,UNIT=disk,
//                   DISP=(OLD,KEEP)
//INOUTD   DD        DSNAME=DATASETD,VOL=SER=111116,DISP=OLD,
//                   UNIT=disk
//INOUTE   DD        DSNAME=DATASETE,VOL=SER=111117,DISP=OLD,
//                   UNIT=disk
//SYSUT3   DD        UNIT=SYSDA,SPACE=(TRK,(1))
//SYSUT4   DD        UNIT=SYSDA,SPACE=(TRK,(1))
//SYSIN    DD        *
            COPY     OUTDD=INOUTA
                     INDD=INOUTE
            SELECT   MEMBER=(MA,MJ)
                     INDD=INOUTC
            EXCLUDE  MEMBER=(MM,MN)
            COPY     O=INOUTB,INDD=INOUTD
                     I=((INOUTC,R),INOUTB)
            COPY     O=INOUTD,I=((INOUTB,R))
            SELECT   MEMBER=MM
/*
```

The control statements are discussed below:

- INOUTA DD defines a partitioned data set (DATASETA). This data set contains three members (MA, MB, and MD) and resides on a disk volume.
- INOUTB DD defines a partitioned data set (DATASETB). This data set resides on a disk volume and contains two members (MA and MJ).
- INOUTC DD defines a partitioned data set (DATASETC), that resides on a disk volume. This data set contains four members (MF, ML, MM, and MN).
- INOUTD DD defines a partitioned data set (DATASETD). This data set resides on a disk volume and contains two members (MM and MP).
- INOUTE DD defines a partitioned data set (DATASETE), that resides on a disk volume. This data set contains three members (MA, MJ and MK).
- SYSUT3 DD defines a temporary spill data set. One track is allocated on a disk volume.
- SYSUT4 DD defines a temporary spill data set. One track is allocated on a disk volume.
- SYSIN DD defines the control data set, which follows in the input stream. The data set contains three COPY statements, two SELECT statements, one EXCLUDE statement, and several INDD statements.
- The first COPY statement indicates the start of a copy operation. The OUTDD operand specifies INOUTA as the DD statement for the output data set (DATASETA).

The first INDD statement specifies INOUTE as the DD statement for the first input data set (DATASETE) to be processed. Processing occurs, as follows:

1. Member MA is searched for and found, but is not copied because the replace option is not specified.

2. Member MJ is searched for, found, and copied to the output data set. Members are not searched for again after they are found.

SELECT specifies the members (MA and MJ) to be selected from the input data set (DATASETE) to be copied.

The second INDD statement marks the end of the first copy step and the beginning of the second copy step within the first copy operation. It specifies INOUTC as the DD statement for the second input data set (DATASETC) to be processed. Members MF and ML, which are not named on the EXCLUDE statement, are copied because neither exists on the output data set. EXCLUDE specifies the members (MM and MN) to be excluded from the second copy operation.

- The second COPY statement indicates the start of another copy operation. The absence of a SELECT or EXCLUDE statement causes a default to a full copy. The O (OUTDD) parameter specifies INOUTB as the output data set (DATASETB). The INDD parameter specifies INOUTD as the first input data set (DATASETD) to be processed. Members MP and MM are copied to the output data set.

  INDD(I) specifies INOUTC as the DD statement for the second input data set (DATASETC) and INOUTB as the DD statement for the third input data set (DATASETB) to be processed. Members MF, ML, MM, and MN are copied from DATASETC. Member MM is copied, although it already exists on the output partitioned data sets, because the replace option is specified. (The pointer in the output data set directory is changed to point to the new (copied) member MM; thus the space occupied by the replaced member MM is embedded, unused space.) Because DATASETB is also the data set specified in the OUTDD parameter, a compress-in-place takes place, and thus the embedded, unused space is removed.

- The third COPY statement indicates the start of another copy operation. The O (OUTDD) parameter specifies INOUTD as the DD statement for the output data set (DATASETD). The I (INDD) parameter specifies INOUTB as the DD statement for the input data set (DATASETB).

  SELECT specifies the member (MM) to be selected from the input partitioned data set (DATASETB) to be copied. The replace option is specified on the data set level.

The temporary spill data sets may or may not be opened, depending on the amount of virtual storage available; therefore, the SYSUT3 and SYSUT4 DD statements should always appear in the job stream.

IEBCOPY EXAMPLE 13

In this example, a partitioned data set (SYS1.LINKLIB) is unloaded to a tape volume.

```
//UNLOAD     JOB   ....
//STEP1      EXEC  PGM=IEBCOPY,PARM='SIZE=100K'
//SYSPRINT   DD    SYSOUT=A
//INPDS      DD    DSNAME=SYS1.LINKLIB,UNIT=disk,DISP=SHR,
//                 VOL=SER=666666
//OUTTAPE    DD    DSNAME=LINKLIB,UNIT=tape,VOL=SER=TAPE00,
//                 LABEL=(,SL),DISP=(NEW,KEEP)
//SYSUT3     DD    DSNAME=TEMP1,UNIT=disk,VOL=SER=111111,
//                 DISP=(NEW,DELETE),SPACE=(80,(60,45))
//SYSIN      DD    *
           COPY  OUTDD=OUTTAPE
                 INDD=INPDS
/*
```

The control statements are discussed below:

- EXEC specifies the execution of IEBCOPY. The PARM parameter specifies the size of the input/output buffer to be used (100K).
- INPDS DD defines a partitioned data set (SYS1.LINKLIB), which resides on a disk volume. This data set is assumed to have 700 members; the number of members is used to calculate the space allocation for SYSUT3.
- OUTTAPE DD defines a sequential data set to which SYS1.LINKLIB is to be unloaded. The unloaded data set is named LINKLIB. If a tape volume is used, it can be IBM standard labeled or unlabeled.
- SYSUT3 DD defines a temporary spill data set on a disk volume. This data set is used if there is not enough space in virtual storage for the input partitioned data set's directory entries. This data set may or may not be opened depending on the amount of virtual storage available; therefore, it is suggested that the statement always appear in the job stream.
- SYSIN DD defines the control data set, which follows in the input stream. The data set contains a COPY and INDD statement.
- COPY indicates the start of an unload operation because the OUTDD parameter refers to OUTTAPE DD, which specifies a sequential output data set. Because no EXCLUDE or SELECT statement is specified, the entire data set is unloaded.
- INDD refers to INPDS DD, which defines the input partitioned data set to be unloaded. Note that for an unload operation, only one INDD data set may be specified for each OUTDD data set.

The SYSUT4 data set is never used for an unload operation. The SYSUT3 data set for an unload operation is used under the same conditions as it is used for a copy operation.

If too much space is allocated with the SIZE option of the PARM parameter on the EXEC statement, the paging process slows down because the buffer areas are fixed.

IEBCOPY EXAMPLE 14

In this example, a sequential data set created by an IEBCOPY unload operation is loaded.

```
//LOAD     JOB  ....
//STEPA    EXEC PGM=IEBCOPY,PARM='SIZE=65536'
//SYSPRINT DD   SYSOUT=A
//SEQIN    DD   DSNAME=UNLOADSET,UNIT=tape,LABEL=(,SL),
//              VOL=SER=TAPE01,DISP=OLD
//INOUT4   DD   DSNAME=DATASET4,UNIT=disk,VOL=SER=2222222,
//              DISP=(NEW,KEEP),SPACE=(CYL,(10,5,10))
//SYSUT3   DD   DSN=TEMP1,UNIT=disk,VOL=SER=111111,
//              DISP=(NEW,DELETE),SPACE=(80,(15,1))
//SYSIN    DD   *
             COPY OUTDD=INOUT4,INDD=SEQIN
/*
```

The control statements are discussed below:

- EXEC specifies the execution of IEBCOPY. The PARM parameter allocates 2 tracks on a disk volume. If less space is specified, 2 tracks are allocated because 2 tracks are the minimum required by IEBCOPY when the unloaded data set's block size does not exceed the track capacity.
- SEQIN DD defines a sequential data set that was previously unloaded by IEBCOPY. The data set contains 28 members in sequential organization.
- INOUT4 DD defines a partitioned data set on a disk volume. This data set is to be kept after the load operation. Ten cylinders are allocated for the data set; ten blocks are allocated for directory entries.
- SYSUT3 DD defines a temporary spill data set on a disk volume. This data set is used if there is not enough space in main storage for the input data set's directory entries. This data set may or may not be opened, depending on the amount of main storage available; therefore, it is suggested that the statement always appear in the job stream. The space allocated for this data set is based on the number of members in the input data set (in this case, 28).
- SYSIN DD defines the control data set, which follows in the input stream. The data set contains a COPY statement.
- COPY indicates the start of a load operation because the INDD parameter refers to SEQIN DD, which defines a sequential data set, and OUTDD refers to INOUT4 DD, which defines a direct access volume.

Because the output data set in this example is new, the SYSUT4 data set is not needed. SYSUT4 should be specified, however, when the output data set is old.

IEBCOPY EXAMPLE 15

In this example, members are selected, excluded, unloaded, loaded, and copied. Processing will occur, as follows: (1) unload, excluding members, (2) unload, selecting members, and (3) load and copy to merge members.

```
//COPY     JOB  ....
//STEP     EXEC PGM=IEBCOPY
//SYSPRINT DD   SYSOUT=A
//PDS1     DD   DSNAME=ACCOUNTA,UNIT=3350,VOL=SER=333333,
//              DISP=OLD
//PDS2     DD   DSNAME=ACCOUNTB,UNIT=3350,VOL=SER=333333,
//              DISP=OLD
//SEQ1     DD   DSNAME=SAVAC,UNIT=3350,VOL=SER=333333,
//              DISP=(NEW,KEEP),SPACE=(CYL,(5,2))
//SEQ2     DD   DSNAME=SAVACB,UNIT=tape,VOL=SER=T01911,
//              DISP=(NEW,KEEP),LABEL=(,SL)
//NEWUP    DD   DSNAME=NEWACC,UNIT=tape,VOL=SER=T01219,
//              DISP=OLD,LABEL=(,SL)
//MERGE    DD   DSNAME=ACCUPDAT,UNIT=3330-1,VOL=SER=222222,
//              DISP=OLD
//SYSUT3   DD   DSNAME=TEMP1,VOL=SER=666666,UNIT=3330-1,
//              DISP=(NEW,DELETE),SPACE=(80,(1,1))
//SYSUT4   DD   DSNAME=TEMP2,VOL=SER=666666,UNIT=3330-1,
//              DISP=(NEW,DELETE),
//              SPACE=(256,(1,1)),DCB=(KEYLEN=8)
//SYSIN    DD   *
     COPY       OUTDD=SEQ1,INDD=PDS1
     EXCLUDE    MEMBER=(D,C)
     COPY       OUTDD=SEQ2,INDD=PDS2
     SELECT     MEMBER=(A,K)
     COPY       OUTDD=MERGE,INDD=((NEWUP,R),PDS1,PDS2)
     EXCLUDE    MEMBER=A
/*
```

The control statements are discussed below:

- PDS1 DD defines a partitioned data set called ACCOUNTA that contains six members (A, B, C, D, E, and F) and resides on a 3350 volume.
- PDS2 DD defines a partitioned data set called ACCOUNTB that contains three members (A, K, and L) and resides on a 3350 volume.
- SEQ1 DD defines a new sequential data set called SAVAC on a 3350 volume.
- SEQ2 DD defines a new sequential data set called SAVACB on a tape volume. The tape has IBM standard labels.
- NEWUP DD defines an old sequential data set called NEWACC that is the unloaded form of a partitioned data set that contains eight members (A, B, C, D, M, N, O, and P). It resides on a tape volume.
- MERGE DD defines a partitioned data set called ACCUPDAT that contains six members (A, B, C, D, Q, and R) and resides on a 3330-1 volume.
- SYSUT3 DD defines a temporary spill data set on a 3330-1 volume.
- SYSUT4 DD defines a temporary spill data set on a 3330-1 volume.
- SYSIN DD defines the control data set, which follows in the input stream.
- The first COPY statement indicates the start of the first unload operation. (The input data set is partitioned; the output data set is sequential.)

- The first EXCLUDE statement specifies that members D and C are to be excluded from the unload operation specified by the preceding COPY statement.
- The second COPY statement indicates the start of the second unload operation. (The input data set is partitioned; the output data set is sequential.)
- The SELECT statement specifies that members A and K are to be included in the unload operation specified by the preceding COPY statement.
- The third COPY statement indicates the start of the copy and load operations. The replace option is specified for the NEWUP data set; therefore, members in this data set replace identically named members on the output data set. The first INDD data set is an unloaded data set that is to be loaded. The second and third INDD data sets are partitioned data sets that are to be copied. (The input data sets are sequential and partitioned; the output data set is partitioned.)
- The second EXCLUDE statement specifies that member A is excluded from the copy and load operation specified in the preceding COPY statement.

IEBCOPY EXAMPLE 16

In this example, all members of data set MODLIBJ, members MODX, MODY, and MODZ of data set MODLIBK, and all members of data set MODLIBL except MYMACRO and MYJCL are altered in place.

```
//ALTERONE JOB  ....
//STEPA    EXEC PGM=IEBCOPY
//SYSPRINT DD   SYSOUT=A
//SYSUT3   DD   UNIT=SYSDA,SPACE=(TRK,(5,1))
//SYSUT4   DD   UNIT=SYSDA,SPACE=(TRK,(5,1))
//LIBJ     DD   DSNAME=MODLIBJ,DISP=(OLD,KEEP)
//LIBK     DD   DSNAME=MODLIBK,DISP=(OLD,KEEP)
//LIBL     DD   DSNAME=MODLIBL,DISP=(OLD,KEEP)
//SYSIN    DD   *
     ALTERMOD  OUTDD=LIBJ
     ALTERMOD  OUTDD=LIBK,LIST=NO
     SELECT    MEMBER=(MODX,MODY,MODZ)
     ALTERMOD  OUTDD=LIBL
     EXCLUDE   MEMBER=(MYMACRO,MYJCL)
/*
```

The control statements are discussed below.

- LIBJ DD defines the partitioned data set MODLIBJ, which has been previously created and cataloged.[2]
- LIBK DD defines the partitioned data set MODLIBK, which has been previously created and cataloged.[2]
- LIBL DD defines the partitioned data set MODLIBL, which has been previously created and cataloged.[2]
- SYSIN DD defines the control data set, which follows in the input stream.

---

[2] For data sets that have not been previously cataloged, you must also specify UNIT and VOL=SER information on the DD statement.

- The first ALTERMOD statement specifies that the entire data set defined in LIBJ is to be altered in place.
- The second ALTERMOD statement plus the following SELECT statement indicates that members MODX, MODY, and MODZ are to be altered in place. The remainder of MODLIBK is unchanged.
- The third ALTERMOD statement plus the following EXCLUDE statement indicates that all of MODLIBL is to be altered in place except the members called MYMACRO and MYJCL. These members remain unchanged.

IEBCOPY EXAMPLE 17

In this example, members MOD7, MOD8, and MOD9 of data set MODLIBL are copied to data set MODLIBM, altered, and reblocked to the default size. All members of data set MODLIBN except NEWMACRO and NEWJCL are copied to data set MODLIBP, altered, and reblocked to 10K bytes; blocks as small as 2K bytes may be written to improve utilization of disk space.

```
                                                                      72
//COPYRBLK JOB  ...
//STEPA    EXEC PGM=IEBCOPY
//SYSPRINT DD   SYSOUT=A
//SYSUT3   DD   UNIT=SYSDA,SPACE=(TRK,(5,1))
//SYSUT4   DD   UNIT=SYSDA,SPACE=(TRK,(5,1))
//LIBL     DD   DSNAME=MODLIBL,DISP=(OLD,KEEP)
//LIBM     DD   DSNAME=MODLIBM,DISP=(OLD,KEEP)
//LIBN     DD   DSNAME=MODLIBN,DISP=(OLD,KEEP)
//LIBP     DD   DSNAME=MODLIBP,DISP=(OLD,KEEP)
//SYSIN    DD   *
     COPYMOD    INDD=LIBL,OUTDD=LIBM
     SELECT     MEMBER=(MOD7,MOD8,MOD9)
     COPYMOD    INDD=LIBN,OUTDD=LIBP,MAXBLK=10K,                      X
                MINBLK=2K,LIST=NO
     EXCLUDE    MEMBER=(NEWMACRO,NEWJCL)
/*
```

The control statements are discussed below.

- LIBL DD defines the partitioned data set MODLIBL, which has been previously created and cataloged.[3]
- LIBM DD defines the partitioned data set MODLIBM, which has been previously created and cataloged.[3]
- LIBN DD defines the partitioned data set MODLIBN, which has been previously created and cataloged.[3]
- LIBP DD defines the partitioned data set MODLIBP, which has been previously created and cataloged.[3]
- SYSIN DD defines the control data set, which follows in the input stream.
- The COPYMOD statement indicates that the members listed in the following SELECT statement (MOD7,MOD8,MOD9) are to be copied from MODLIBL to MODLIBM, altered, and reblocked.
- The second COPYMOD statement indicates that the MODLIBN data set (except for NEWMACRO and NEWJCL, which are specified in

[3] For data sets that have not been previously cataloged, you must also specify UNIT and VOL=SER information on the DD statement.

the following EXCLUDE statement) is copied to MODLIBP, altered, and reblocked to 10K bytes.

IEBCOPY EXAMPLE 18

In this example, all members of data set MODLIBY are copied to tape COPYLIBY in STEP1. MODLIBY is scratched (but not uncataloged) in STEP2. In STEP3, all members are copied back to data set MODLIBY, reblocked to the default size, and altered. The net result is that the data set MODLIBY is compressed, altered, and reblocked.

```
//COPYTWO  JOB  ...
//STEP1    EXEC PGM=IEBCOPY
//SYSPRINT DD   SYSOUT=A
//SYSUT3   DD   UNIT=SYSDA,SPACE=(TRK,(5,1))
//SYSUT4   DD   UNIT=SYSDA,SPACE=(TRK,(5,1))
//LIBY     DD   DSNAME=MODLIBY,DISP=(OLD),
//              UNIT=3330-1,VOL=SER=333101
//TAPEA    DD   DSNAME=COPYLIBY,DISP=(NEW,PASS),
//              UNIT=tape,VOL=SER=717000,LABEL=(,NL)
//SYSIN    DD   *
     COPY INDD=LIBY,OUTDD=TAPEA
/*
//STEP2    EXEC PGM=IEHPROGM,COND=(0,EQ,STEP1)
//SYSPRINT DD   SYSOUT=A
//LIBY     DD   DSNAME=MODLIBY,DISP=(OLD),
//              UNIT=3330-1,VOL=SER=333101
//SYSIN    DD   *
     SCRATCH DSNAME=MODLIBY,VOL=3330-1=333101
/*
//STEP3    EXEC PGM=IEBCOPY,COND=(0,EQ,STEP1)
//SYSPRINT DD   SYSOUT=A
//SYSUT3   DD   UNIT=SYSDA,SPACE=(TRK,(5,1))
//SYSUT4   DD   UNIT=SYSDA,SPACE=(TRK,(5,1))
//LIBY     DD   DSNAME=MODLIBY,DISP=(NEW,KEEP),
//              UNIT=3330-1,VOL=SER=333101,
//              SPACE=(TRK,(2,1,5)
//TAPEA    DD   DSNAME=COPYLIBY,DISP=(OLD,KEEP),
//              UNIT=tape,VOL=SER=717000,LABEL=(,NL)
//SYSIN    DD   *
     COPYMOD INDD=TAPEA,OUTDD=LIBY
/*
```

The control statements are discussed below.

- STEP1 marks the beginning of the IEBCOPY job step.
- LIBY DD defines the partitioned data set MODLIBY, which has also been previously defined.
- TAPEA DD defines the tape data set COPYLIBY.
- The COPY statement makes a backup copy of MODLIBY and places it in the data set COPYLIBY.
- STEP2 marks the beginning of the IEHPROGM job step. If STEP1 fails, STEP2 will not be executed.
- The SCRATCH statement scratches the old data set MODLIBY but does not remove it from the catalog.
- STEP3 marks the beginning of the second IEBCOPY job step. STEP3 will not be executed if STEP1 fails.
- The COPYMOD statement copies all members back to MODLIBY, alters their RLD counts, and reblocks them. The new MODLIBY will be compressed, but will not necessarily occupy the same

space on the disk as it did before being scratched and reallocated.

IEBDG PROGRAM

IEBDG is a data set utility used to provide a **pattern** of test data to be used as a programming debugging aid.

An output data set, containing records of any format, can be created through the use of utility control statements, with or without input data. An optional user exit passes control to a user routine to monitor each output record before it is written. Sequential, ISAM, and partitioned data sets can be used for input or output.

You can code utility control statements to generate a pattern of data that can be analyzed quickly for predictable results.

When you define the contents of a field, the following must be decided:

- What type of pattern—IBM-supplied or user-supplied—is to be placed initially in the defined field.
- What action, if any, is to be performed to alter the contents of the field after it is selected for each output record.

TYPES OF PATTERNS

IBM-SUPPLIED PATTERNS

IBM supplies seven patterns:

- Alphameric
- Alphabetic
- Zoned decimal
- Packed decimal
- Binary number
- Collating sequence
- Random number

You may choose one of them when defining the contents of a field. All patterns except the binary and random number patterns repeat in a given field, provided that the defined field length is sufficient to permit repetition. For example, the alphabetic pattern is:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFG...
```

Figure 34 on page 98 shows the IBM-supplied patterns.

Type	Expressed in Hexadecimal	Expressed in Printable Characters
Alphameric	C1 C2...E9, F0...F9	AB...Z, 0...9
Alphabetic	C1 C2...E9	AB...Z
Zoned Decimal	F0F0...F9F9	00...99
Packed Decimal	0000...001C (Positive pattern) 0000...001D (Negative pattern)	Not applicable
Binary Number	00000001, etc. (Positive pattern) FFFFFFFF, etc. (Negative pattern)	Not applicable
Collating Sequence	40...F9	b¢.<(+|&!$*);¬-/,%_>?:#=" A...Z 0...9
Random Number	Random hexadecimal digits	Not applicable

Figure 34. IBM-Supplied Patterns

A packed decimal or binary number is right-aligned in the defined field.

You can specify a starting character when defining an alphameric, alphabetic, or collating-sequence field. For example, a 10-byte alphabetic field for which "H" is specified as the starting character would appear as:

```
HIJKLMNOPQ
```

The same 10-byte alphabetic field with no specified starting character would appear as:

```
ABCDEFGHIJ
```

You can specify a mathematical sign when defining a packed decimal or binary field. If no sign is specified, the field is assumed to be positive.

USER-SPECIFIED PICTURES

Instead of selecting an IBM-supplied pattern, you may wish to specify a picture to be placed in the defined field. The user can provide:

- An EBCDIC character string
- A decimal number to be converted to packed decimal by IEBDG
- A decimal number to be converted to binary by IEBDG

When you supply a picture, a picture length must be specified that is equal to or less than the specified field length. An EBCDIC picture is left-aligned in a defined field; a decimal number that is converted to packed decimal or to binary is right-aligned in a defined field.

You can initially load (fill) a defined field with either an EBCDIC character or a hexadecimal digit. For example, the 10-byte picture "BADCFEHGJI" is to be placed in a 15-byte field. An EBCDIC "2" is to be used to pad the field. The result is BADCFEHGJI22222. (If no fill character is provided, the remaining bytes contain binary zeros.) Remember that the fill character, if specified, is written in each byte of the defined field prior to the inclusion of an IBM-supplied pattern or user-supplied picture.

MODIFICATION OF SELECTED FIELDS

IEBDG can be used to change the contents of a field in a specified manner. One of eight actions can be selected to change a field after its inclusion in each applicable output record. These actions are:

- Ripple
- Shift left
- Shift right
- Truncate left
- Truncate right
- Fixed
- Roll
- Wave

Figure 35 shows the effects of each of the actions on a 6-byte alphabetic field. Note that the roll and wave actions are applicable only when a user pattern is supplied. In addition, the result of a ripple action depends on which type of pattern—IBM-supplied or user-supplied—is present.

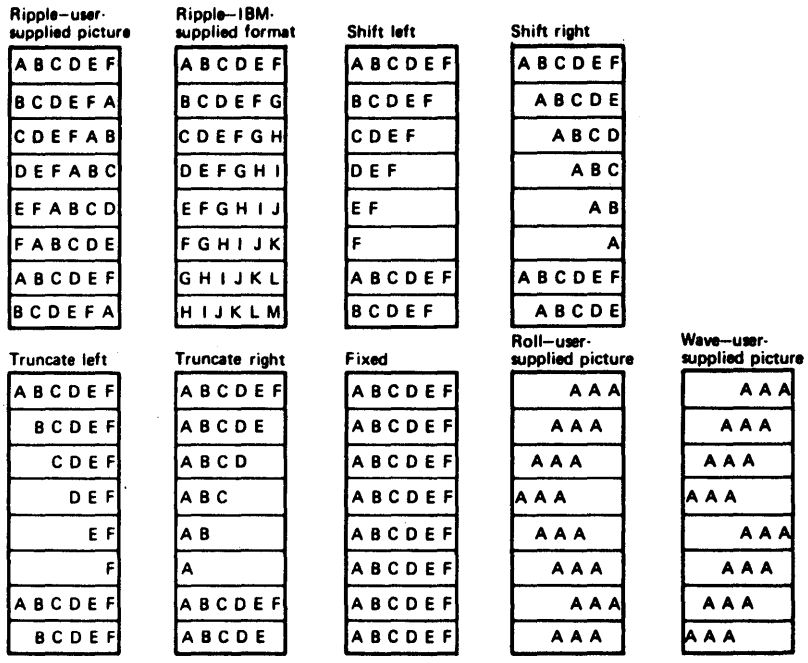


Figure 35. IEBDG Actions

If no action is selected, or if the specified action is not compatible with the format, the **fixed** action is assumed by IEBDG.

INPUT AND OUTPUT

IEBDG uses the following input:

- An input data set which contains records to be used in the construction of an output data set or partitioned data set member. The input data sets are optional; that is, output records can be created entirely from utility control statements.
- A control data set that contains any number of sets of utility control statements.

IEBDG produces the following output:

- An output data set that is the result of the IEBDG operation. One output data set is created by each set of utility control statements included in the job step.
- A message data set that contains informational messages, the contents of applicable utility control statements, and any error messages.

Input and output data sets may be sequential, indexed sequential (ISAM), or partitioned data set members. BDAM and VSAM are not supported.

RETURN CODES

IEBDG returns a code in register 15 to indicate the results of program execution. The return codes and their meanings are listed below.

Codes	Meaning
00 (00 hex)	Successful completion.
04 (04)	A user routine returned a code of 16 to IEBDG. The job step is terminated at the user's request.
08 (08)	An error occurred while processing a set of utility control statements. No data is generated following the error. Processing continues normally with the next set of utility control statements, if any.
12 (0C)	An error occurred while processing an input or output data set. The job step is terminated.
16 (10)	An error occurred from which recovery is not possible. The job step is terminated.

Figure 36. IEBDG Return Codes

CONTROL

IEBDG is controlled by job control statements and utility control statements. The job control statements are used to execute or invoke IEBDG and define the data sets used and produced by IEBDG. Utility control statements are used to control the functions of the program and to define the contents of the output records.

JOB CONTROL STATEMENTS

Figure 37 shows the job control statements for IEBDG.

Both input and output data sets can contain fixed, variable, or undefined records.

Statement	Use
JOB	Initiates the job.
EXEC	Specifies the program name (PGM=IEBDG) or, if the job control statements reside in a procedure library, the procedure name. Additional information can be specified in the EXEC statement; see "PARM Information on the EXEC Statement" on page 102.
SYSPRINT DD	Defines a sequential message data set. The data set can be written on a system output device, a tape volume, or a DASD volume.
SYSIN DD	Defines the control data set, which contains the utility control statements and, optionally, input records. The data set normally resides in the input stream; however, it can be defined as a sequential data set or as a member of a partitioned data set.
seqinset DD	Defines an optional sequential or ISAM data set used as input to IEBDG. The data set can reside on a tape volume or on a DASD volume. Any number of these statements (each having a ddname different from all other ddnames in the job step) can be included in the job step. Each DD statement is subsequently referred to by a DSD utility control statement.
parinset DD	Defines an optional input partitioned data set member residing on a DASD volume. Any number of these statements (each having a ddname different from all other ddnames in the job step) can be included in the job step. The DD statement is subsequently referred to by a DSD utility control statement.
seqout DD	Defines an output (test) sequential or ISAM data set. Any number of these DD statements can be included per job step; however, only one statement is applicable per set of utility control statements.
parout DD	Defines an optional output partitioned data set member to be created and placed on a DASD volume. Any number of these DD statements (each DD statement referring to the same or to a different data set) can be included per job step; however, only one statement is applicable per set of utility control statements.

Figure 37. Job Control Statements for IEBDG

The DSORG subparameter must be included in the DCB subparameters if the input or output data set has an indexed sequential (ISAM) organization (DSORG=IS). If members of a partitioned data set are used, DSORG=PO or DSORG=PS may be coded. If the DSORG subparameter is not coded, DSORG=PS is assumed.

For an ISAM data set, the key length must be specified in the DCB.

Refer to Data Management Services for information on estimating space allocations.

PARM Information on the EXEC Statement

The EXEC statement can include an optional PARM parameter to specify the number of lines to be printed between headings in the message data set, coded as follows:

PARM=LINECT=nnnn

The nnnn is a 4-digit decimal number that specifies the number of lines (0000 to 9999) to be printed per page of output listing.

If PARM is omitted, 58 lines are printed between headings (unless a channel 12 punch is encountered in the carriage control tape, in which case a skip to channel 1 is performed and a heading is printed).

If IEBDG is invoked, the line-count option can be passed in a parameter list that is referred to by a subparameter of the LINK or ATTACH macro instruction. In addition, a page count can be passed in a six-byte parameter list that is referred to by a subparameter of the LINK or ATTACH macro instruction. For a discussion of linkage conventions, refer to "Invoking Utility Programs from a Problem Program" on page 13.

SYSPRINT DD Statement

If the SYSPRINT DD statement is omitted, no messages are written. The block size for the SYSPRINT data set must be a multiple of 121. Any blocking factor can be specified.

SYSIN DD Statement

The block size for the SYSIN data set must be a multiple of 80. Any blocking factor can be specified.

seqinset DD Statement

The "seqinset" DD statement can be entered:

```
//seqinset DD DSNAME=setname,UNIT=xxxx,DISP=(OLD,KEEP),
//            VOLUME=SER=xxxxxx,LABEL=(...,...),
//            DCB=(applicable subparameters)
```

The LABEL parameter is included only for a magnetic tape volume. If the input data set has an indexed sequential organization, DSORG=IS should be coded in the DCB parameter.

parinset DD Statement

The "parinset" DD statement can be entered:

```
//parinset DD DSNAME=setname(membername),UNIT=xxxx,
//            DISP=(OLD,KEEP),VOLUME=SER=xxxxxx,
//            DCB=(applicable subparameters)
```

seqout DD Statement

The "seqout" DD statement can be entered:

```
//seqout DD DSNAME=setname, UNIT=xxxx,
//          DISP=(,KEEP),VOLUME=SER=xxxxxx,
//          DCB=(applicable subparameters)
```

The LABEL parameter is included for magnetic tape; the SPACE parameter is included for DASD.

parout DD Statement

The "parout" DD statement can be entered:

```
//parout DD DSNAME=setname(membername),UNIT=xxxx,
//          DISP=(,KEEP),VOLUME=SER=xxxxxx,DCB=(applicable
//          DCB=(applicable subparameters),
//          SPACE=(applicable subparameter)
```

The SPACE parameter is included on the parout DD statement when creating the first member to be placed in a partitioned data set.

The partitioned data set defined by "parout" is a new member and has a new directory entry. No information is copied from the previous directory.

UTILITY CONTROL STATEMENTS

IEBDG is controlled by the following utility control statements:

Statement	Use
DSD	Specifies the ddnames of the input and output data sets. One DSD statement must be included for each set of utility control statements.
FD	Defines the contents and lengths of fields to be used in creating output records.
CREATE	Defines the contents of output records.
REPEAT	Specifies the number of times a CREATE statement or a group of CREATE statements are to be used in generating output records.
END	Marks the end of a set of IEBDG utility control statements.

Figure 38. IEBDG Utility Control Statements

Any number of sets of control statements can appear in a single job step. Each set defines one data set.

General continuation requirements for utility control statements are described in "Continuing Utility Control Statements" on page 5.

FD or CREATE utility control statements that contain a PICTURE parameter and are to be continued must have a nonblank character in column 72. The continuation must begin in column 4 on the next statement.

DSD Statement

The DSD statement marks the beginning of a set of utility control statements and specifies the data sets that IEBDG is to use as input. The DSD statement can be used to specify one output data set and any number of input data sets for each application of IEBDG.

The format of the DSD statement is:

[label]	**DSD**	**OUTPUT=(**ddname**)** **[,INPUT=(**ddname,...**)]**

The ddname SYSIN must **not** be coded in the INPUT parameter.

Each parameter should appear no more than once on any DSD statement.

FD Statement

The FD statement defines the contents and length of a field that will be used subsequently by a CREATE statement (or statements) to form output records. A defined field within the input logical record may be selected for use in the output records if it is referred to, by name, by a subsequent CREATE statement.

Figure 39 shows how fields defined in FD statements are placed in buffer areas so that subsequent CREATE statements can assign selected fields to specific output records.

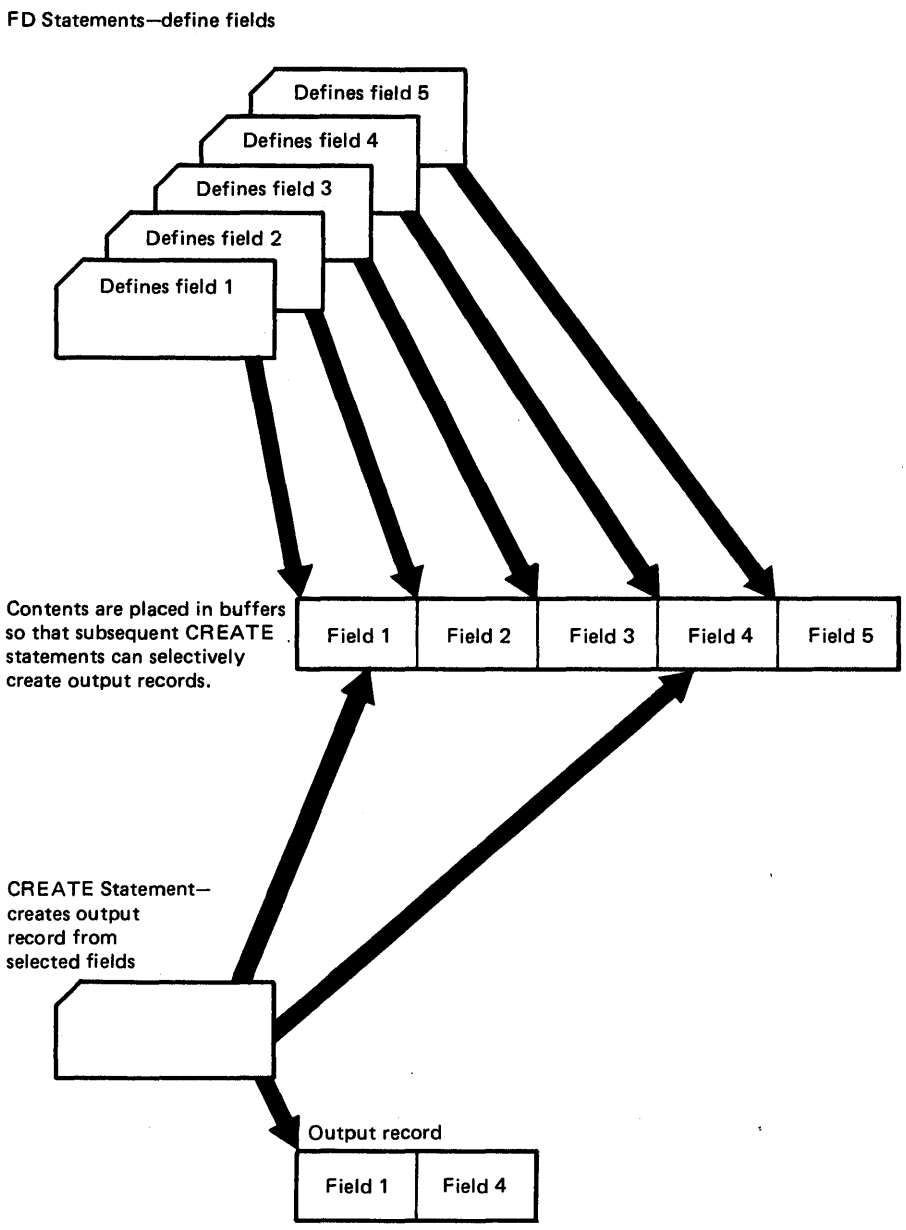


Figure 39. Defining and Selecting Fields for Output Records Using IEBDG

Figure 40 on page 106 shows how the FD statement is used to specify a field in an input record to be used in output records. The left-hand side of the figure shows that a field in the input record beginning at byte 50 is selected for use in the output record. The right-hand side of the figure shows that the field is to be placed at byte 20 in the output record.

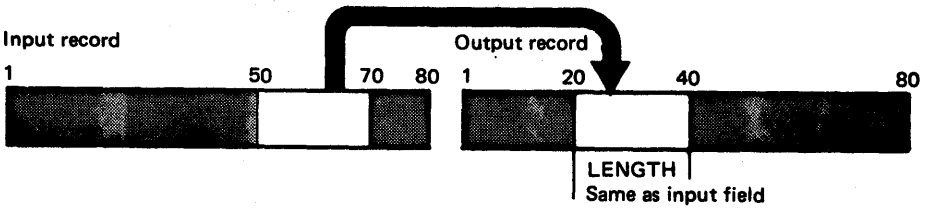


Figure 40. Field Selected from the Input Record for Use in the Output Record

The format of the FD statement is:

[label]	FD	NAME=name ,LENGTH=length-in-bytes [,STARTLOC=starting-byte-location] [,FILL={'character'|X'2-hex-digits'}] [,FORMAT=pattern[,CHARACTER=character]| ,PICTURE=length,{'character-string'| P'decimal-number'| B'decimal-number'}] [,SIGN=sign] [,ACTION=action] [,INDEX=number[,CYCLE=number][,RANGE=number]] [,INPUT=ddname] [,FROMLOC=number]

Some of the FD keywords do not apply when certain patterns or pictures are selected by the user; for example, the INDEX, CYCLE, RANGE, and SIGN parameters are used only with numeric fields. Figure 41 on page 107 shows which IEBDG keywords can be used with the applicable pattern or picture chosen by the user. Each keyword should appear no more than once on any FD statement.

FORMAT/PICTURE Value	Compatible Parameters
FORMAT=AL (alphabetic) FORMAT=AN (alphameric) FORMAT=CO (collating seq.)	ACTION=SL (shift left) ACTION=SR (shift right) ACTION=TL (truncate left) ACTION=TR (truncate right) ACTION=FX (fixed) ACTION=RP (ripple)
FORMAT=ZD (zoned decimal) FORMAT=PD (packed decimal) FORMAT=BI (binary)	INDEX=x CYCLE=x RANGE=x SIGN=x[1]
PICTURE=P'n' (packed decimal) PICTURE=B'n' (binary)	INDEX=x CYCLE=x RANGE=x SIGN=x[1]
PICTURE='string' (EBCDIC)	ACTION=SL (shift left) ACTION=SR (shift right) ACTION=TL (truncate left) ACTION=TR (truncate right) ACTION=FX (fixed) ACTION=RP (ripple) ACTION=WV (wave) ACTION=RO (roll)

Figure 41. Compatible IEBDG Operations

Note to Figure 41:

[1] Zoned decimal numbers (ZD) do not include a sign.

CREATE Statement

The CREATE statement defines the contents of a record (or records) to be made available to a user routine or to be written directly as an output record (or records).

The format of the CREATE statement is:

[label]	CREATE	[QUANTITY=number] [,FILL={'character'|X'2-hex-digits'}] [,INPUT=ddname|SYSIN[(cccc)]] [,PICTURE=length,startloc[,'character-string'| ,P'decimal-number'| ,B'decimal-number'}] [,NAME=name|(name1,namen...)| (name[COPY=number,name1,namen...),...)] [,EXIT=routinename]

After processing each potential output record, the user routine should provide a return code in register 15 to instruct IEBDG how to handle the output record. The user codes are listed below.

Codes	Meaning
00 (00 hex)	The record is to be written.
04 (04)	The record is not to be written. The skipped record is not to be counted as a generated output record; processing is to continue as though a record were written. If skips are requested through user exits and input records are supplied, each skip causes an additional input record to be processed in the generation of output records. For example, if a CREATE statement specifies that 10 output records are to be generated and a user exit indicates that two records are to be skipped, 12 input records are processed.
12 (0C)	The processing of the remainder of this set of utility control statements is to be bypassed. Processing is to continue with the next DSD statement.
16 (10)	All processing is to halt.

Figure 42. IEBDG User Exit Return Codes

When an exit routine is loaded and you return control to IEBDG, register 1 contains the address of the first byte of the output record. Each keyword should appear no more than once on any CREATE statement.

Figure 43 shows the addition of field X to two different records. In record 1, field X is the first field referred to by the CREATE statement; therefore, field X begins in the first byte of the output record. In record 2, two fields, field A and field B, have already been referred to by a CREATE statement; field X, the next field referred to, begins immediately after field B. Field X does not have a special starting location in this example.

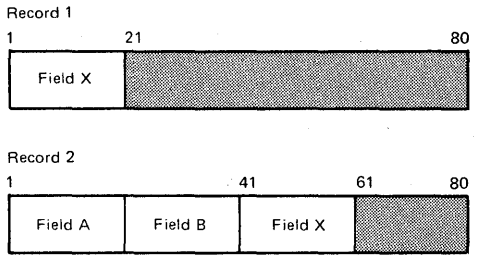


Figure 43. Default Placement of Fields within an Output Record Using IEBDG

You can also indicate that a numeric field is to be modified after it has been referred to n times by a CREATE statement or statements, that is, after n cycles, a modification is to be made. A modification will add a user-specified number to a field.

The CREATE statement constructs an output record by referring to previously defined fields by name and/or by providing a picture to be placed in the record. You can generate multiple records with a single CREATE statement.

When defining a picture in a CREATE statement, the user must specify its length and starting location in the output record. The specified length must be equal to the number of specified EBCDIC or numeric characters. (When a specified decimal number is converted to packed decimal or binary, it is automatically right-aligned.)

Figure 44 shows three ways in which output records can be created from utility control statements.

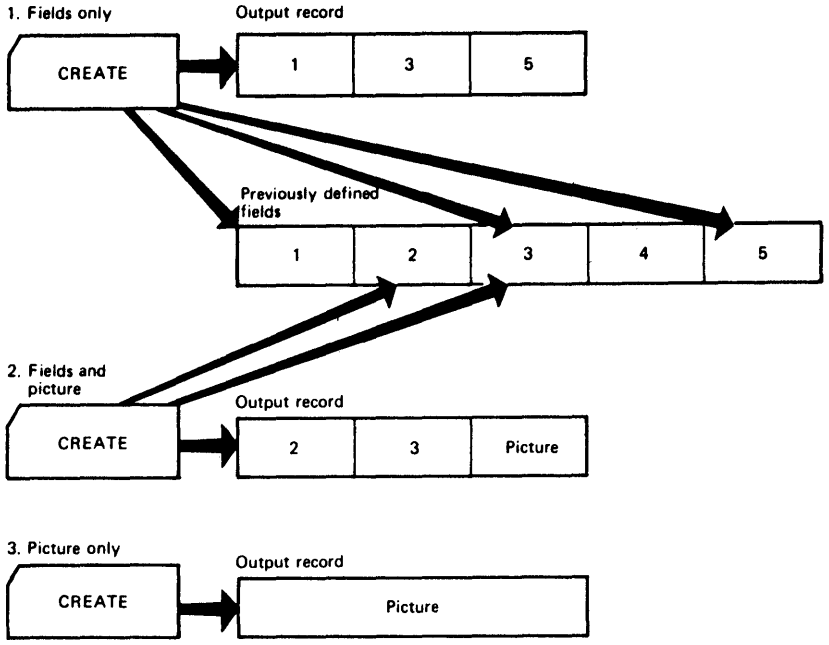


Figure 44. Creating Output Records with Utility Control Statements

As an alternative to creating output records from utility control statements alone, you can provide input records, which can be modified and written as output records. Input records can be provided directly in the input stream, or in a separate data set. Only one input data set can be read for each CREATE statement.

As previously mentioned, the CREATE statement is responsible for the construction of an output record. An output record is constructed in the following order:

1. A fill character, specified or default (binary zero), is initially loaded into each byte of the output record.
2. If the INPUT operand is specified on the CREATE statement, and not on an FD statement, the input records are left-aligned in the corresponding output record.
3. If the INPUT operand specifies a ddname in any FD statement, only the fields described by the FD statement(s) are placed in the output record.

4. FD fields, if any, are placed in the output record in the order of the appearance of their names in the CREATE statement.

5. A CREATE statement picture, if any, is placed in the output record.

IEBDG provides a user exit so you can provide your own routine to analyze or further modify a newly constructed record before it is placed in the output data set. See Appendix A, "Exit Routine Linkage" on page 438 for information on linking to a user exit routine.

A set of utility control statements contains one DSD statement, any number of FD, CREATE, and REPEAT statements, and one END statement when the INPUT parameter is omitted from the FD card.

When selecting fields from an input record (FD INPUT=ddname), the field must be defined by an FD statement within each set of utility control statements. In that case, defined fields for field selection are not usable across sets of utility control statements; such an FD card may be duplicated and used in more than one set of utility control statements within the job step.

REPEAT Statement

The REPEAT statement specifies the number of times a CREATE statement or group of CREATE statements is to be used repetitively in the generation of output records. The REPEAT statement **precedes** the CREATE statements to which it applies.

Figure 45 shows a group of five CREATE statements repeated n times.

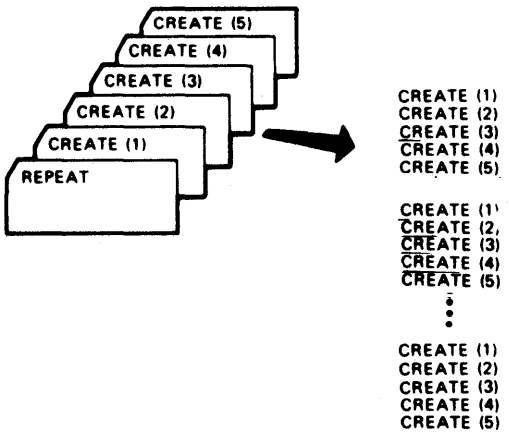


Figure 45. Repetition Caused by the REPEAT Statement Using IEBDG

The format of the REPEAT statement is:

[label]	REPEAT	QUANTITY=number[,CREATE=number]

END Statement

The END statement is used to mark the end of a set of utility control statements. Each set of control statements can pertain

to any number of input data sets but only a single output data set.

The format of the END statement is:

[label]	END

Parameters	Applicable Control Statements	Description of Parameters
ACTION	FD	**ACTION=**<u>action</u> specifies how the contents of a defined field are to be altered (if at all) after the field's inclusion in an output record. These values can be coded: **FX** specifies that the contents of a defined field are to remain fixed after the field's inclusion in an output record. **RO** specifies that the contents of a defined field are to be rolled after the field's inclusion in an output record. The picture is incremented to the left by one byte for each output record, until the first non-blank character of the picture is in field byte 1. At that time, the character string is reset to its original picture position. RO can be used only for a user-defined field. For RO to be effective, the picture length must be less than the field length. **RP** specifies that the contents of a defined field are to be rippled after the field's inclusion in an output record. **SL** specifies that the contents of a defined field are to be shifted left after the field's inclusion in an output record. **SR** specifies that the contents of a defined field are to be shifted right after the field's inclusion in an output record. **TL** specifies that the contents of a defined field are to be truncated left after the field's inclusion in an output record. **TR** specifies that the contents of a defined field are to be truncated right after the field's inclusion in an output record.

Parameters	Applicable Control Statements	Description of Parameters
ACTION (continued)	FD (continued)	**WV** specifies that the contents of a defined field are to be waved after the field's inclusion in an output record. The picture is incremented to the left by one byte for each output record, until the first non-blank character of the picture is in field byte 1. At this time, the character string is reset to its original picture position. WV can be used only for a user-defined field. For WV to be effective, the picture length must be less than the field length. **Default:** FX See Figure 41 on page 107 for system actions compatible with FORMAT and PICTURE values. See Figure 35 on page 99 for examples of IEBDG ACTION patterns.
CREATE	REPEAT	**CREATE=**number specifies the number of following CREATE statements to be included in the group. **Default:** Only the first CREATE statement is repeated.
EXIT	CREATE	**EXIT=**routinename specifies the name of the user routine that is to receive control from IEBDG before writing each output record.
FILL	CREATE FD	**FILL={**'character'|X'2-hex-digits'**}** specifies a value that is to be placed in each byte of the output record before any other operation in the construction of record. These values can be coded: 'character' specifies an EBCDIC character that is to be placed in each byte of the output record. X'2-hex-digits' specifies 2 hexadecimal digits (for example, FILL=X'40', or FILL=X'FF') to be placed in each byte of the output record. **Default:** Binary zeros are placed in the output record.

Parameters	Applicable Control Statements	Description of Parameters
FORMAT	FD	**FORMAT=**pattern[,**CHARACTER=**character] specifies an IBM-supplied pattern that is to be placed in the defined field. FORMAT must not be used when PICTURE is used. The values that can be coded are: pattern specifies the IBM-supplied patterns, as follows: **AL** specifies an alphabetic pattern. **AN** specifies an alphameric pattern. **BI** specifies a binary pattern. **CO** specifies a collating sequence pattern. **PD** specifies a packed decimal pattern. **RA** specifies a random binary pattern. **ZD** specifies a zoned decimal pattern. **CHARACTER=**character specifies the starting character of a field. See "IBM-Supplied Patterns" on page 97 for details on starting characters.
FROMLOC	FD	**FROMLOC=**number specifies the location of the selected field within the input logical record. The number represents the position in the input record. If, for example, FROMLOC=10 is coded, the specified field begins at the tenth byte; if FROMLOC=1 is coded, the specified field begins at the first byte. (For variable-length records, significant data begins on the first byte after the 4-byte length descriptor.) When retrieving data sets with RECFM=F or FB, and RKP>0, the record consists of the key plus the data with embedded key. To copy the entire record, the output DCB=LRECL has to be input LRECL + KEYLEN. If only the data (which includes the embedded key) is to be copied, the FROMLOC must point to start of the data, that is, FROMLOC=keylength. **Default:** The start of the input record.

Parameters	Applicable Control Statements	Description of Parameters
INDEX	FD	**INDEX=**number**[,CYCLE=**number**][,RANGE=**number**]** specifies a decimal number to be added to this field whenever a specified number of records have been written. INDEX is valid only with FORMATs ZD, PD, BI, or PICTURES P'n', B'n'. Additional values can be coded: **CYCLE=**number specifies a number of output records (to be written as output or made available to an exit routine) that are treated as a group by the INDEX keyword. Whenever this field has been used in the construction of the specified number of records, it is modified as specified in the INDEX parameter. For example, if CYCLE=3 is coded, output records might appear as 111 222 333 444 etc. This parameter can be coded only when INDEX is coded. **RANGE=**number specifies an absolute value which the contents of this field can never exceed. If an index operation attempts to exceed the specified absolute value, the contents of the field as of the previous index operation are used. **Default:** No indexing is performed. If CYCLE is omitted and INDEX is coded, a CYCLE value of 1 is assumed; that is, the field is indexed after each inclusion in a potential output record.

Parameters	Applicable Control Statements	Description of Parameters
INPUT	DSD	**INPUT=(**ddname,...**)** specifies the ddname of a DD statement defining a data set used as input to the program. Any number of data sets can be included as input—that is, any number of ddnames referring to corresponding DD statements can be coded. Whenever ddnames are included on a continuation card, they must begin in column 4. The ddname SYSIN must not be coded as the INPUT parameter on the DSD control statement. Each ddname should not appear more than once on any control statement.
	FD	**INPUT=**ddname specifies the ddname of a DD statement defining a data set used as input for field selection. Only a portion of the record described by the FD statement will be placed in the output record. If the record format of the output data set indicates variable-length records, the position within the output record will depend upon where the last insert into the output record was made unless STARTLOC is specified. The ddname SYSIN must not be coded as the INPUT parameter on the FD control statement. Each ddname should not appear more than once on any control statement. A corresponding ddname must also be specified in the associated CREATE statement in order to have the input record(s) read.

Parameters	Applicable Control Statements	Description of Parameters
INPUT (continued)	CREATE	**INPUT=**ddname|**SYSIN**[(cccc)] defines an input data set whose records are to be used in the construction of output records. If INPUT is coded, QUANTITY should also be coded, unless the remainder of the input records are all to be processed by this CREATE statement. If INPUT is specified in an FD statement referenced by this CREATE statement, there must be a corresponding ddname specified in the CREATE statement in order to get the input record(s) read. These values can be coded: ddname specifies the ddname of a DD statement defining an input data set. **SYSIN**[cccc] specifies that the SYSIN data set (input stream) contains records (other than utility control statements) to be used in the construction of output records. If **SYSIN** is coded, the input records follow this CREATE statement (unless the CREATE statement is in a REPEAT group, in which case the input records follow the last CREATE statement of the group). cccc can be any combination of from 1 to 4 EBCDIC characters. If cccc is coded, the input records are delimited by a record containing EBCDIC characters beginning in column 1. When INPUT=SYSIN with no cccc value, the input records are delimited from any additional utility control statements by a record containing $$$E in columns 1 through 4.
LENGTH	FD	**LENGTH=**length-in-bytes specifies the length in bytes of the defined field. For variable records, 4 bytes of length descriptor must be added. For ACTION=RP or WV, the length is limited to 16383 bytes. For ACTION=R0, the length is limited to 10922 bytes.

Parameters	Applicable Control Statements	Description of Parameters
NAME	FD	**NAME=**name specifies the name of the field defined by this FD statement.
	CREATE	**NAME=**name|(name1,namen...)|(name,(**COPY=**number,name1,namen...)...) specifies the name or names of previously defined fields to be included in the applicable output records. If both NAME and PICTURE are omitted, the fill character specified in the CREATE statement appears in each byte of the applicable output record. These values can be coded: (name1,...) specifies the name or names of a field or fields to be included in the applicable output record(s). Each field (previously defined in the named FD statement) is included in an output record in the order in which its name is encountered in the CREATE statement. **COPY=**number indicates that all fields named in the inner parentheses (maximum of 20) are to be treated as a group and included the specified number of times in each output record produced by this CREATE statement. Any number of sets of inner parentheses can be included with NAME. Within each set of inner parentheses, COPY must appear before the name of any field.
OUTPUT	DSD	**OUTPUT=**(ddname) specifies the ddname of the DD statement defining the output data set.

Parameters	Applicable Control Statements	Description of Parameters
PICTURE	FD CREATE	**PICTURE=**length[,startloc]{,'character-string'| ,P'decimal-number'|,B'decimal-number'} specifies the length, starting byte (CREATE only), and the contents of a user-supplied picture. For FD, PICTURE must not be used when FORMAT is used. If both PICTURE and NAME are omitted, the fill character specified in the CREATE statement appears in each byte of applicable output records. These values can be coded: length specifies the number of bytes that the picture will occupy. length must be equal to or less than the LENGTH parameter value in the FD statement. startloc **(CREATE only)** specifies a starting byte (within any applicable output record) in which the picture is to begin. 'character-string' specifies an EBCDIC character string that is to be placed in the applicable record(s). The character string is left-aligned at the defined starting byte. A character string may be broken in column 71, a non-blank character in column 72 is required, and it must be continued in column 4 of the next statement. The number of characters within the quotation marks must equal the number specified in the length subparameter (for FD statements). **P** 'decimal-number' specifies a decimal number that is to be converted to packed decimal and right-aligned (within the boundaries of the defined length and starting byte) in the output records or defined field. The number of characters within the quotation marks must equal the number specified in the length subparameter (for FD statements). **B** 'decimal-number' specifies a decimal number that is to be converted to binary and right-aligned (within the boundaries of the defined length and starting byte) in the output records or defined field. The number of characters within the quotation marks must equal the number specified in the length subparameter (for FD statements).

Parameters	Applicable Control Statements	Description of Parameters
QUANTITY	CREATE	**QUANTITY=**number specifies the number of records that this CREATE statement is to generate; the contents of each record are specified by the other parameters. If both QUANTITY and INPUT are coded, and the quantity specified is greater than the number of records in the input data set, the number of records created is equal to the number of input records to be processed plus the generated data up to the specified number. **Default:** If QUANTITY is omitted and INPUT is not specified, only one output record is created. If QUANTITY is omitted and INPUT is specified, the number of records created is equal to the number of records in the input data set. If both QUANTITY and INPUT are coded, but the QUANTITY is less than the number of records in the input data set, then only the number of records specified by QUANTITY are written to the output data set.
	REPEAT	**QUANTITY=**number specifies the number of times the defined group of CREATE statements is to be used repetitively. This number cannot exceed 65,535.
SIGN	FD	**SIGN=**sign specifies a mathematical sign (+ or -), to be used when defining a packed decimal or binary field. **Default:** Positive (+).
STARTLOC	FD	**STARTLOC=**starting-byte-location specifies a starting location (within all output records using this field) in which a field is to begin. For example, if the first byte of an output record is chosen as the starting location, the keyword is coded STARTLOC=1; if the tenth byte is chosen, STARTLOC=10 is coded, etc. **Default:** The field will begin in the first available byte of the output record (determined by the order of specified field names in the applicable CREATE statement). For variable records the starting location is the first byte after the length descriptor.

IEBDG EXAMPLES

The following examples illustrate some of the uses of IEBDG. Figure 46 can be used as a quick reference guide to IEBDG examples. The numbers in the "Example" column refer to examples that follow.

Operation	Data Set Organization	Device	Comments	Example
Place binary zeros in selected fields.	Sequential	9-track Tape	Blocked input and output.	1
Ripple alphabetic pattern	Sequential	9-track Tape, Disk	Blocked input and output.	2
Create output records from utility control statements	Sequential	Disk	Blocked output.	3
Modify records from partitioned members and input stream	Partitioned, Sequential	Disk	Reblocking is performed. Each block of output records contains ten modified partitioned input records and two input stream records.	4
Create partitioned members for utility control statements	Partitioned	Disk	Blocked output. One set of utility control statements per member.	5
Roll and wave user-supplied patterns	Sequential	Disk	Output records are created from utility control statements.	6
Create indexed sequential data set using field selection and data generation	Sequential, Indexed Sequential	Disk Tape	Output records are created by augmenting selected input fields with generated data.	7

Figure 46. IEBDG Example Directory

Examples that use **disk** or **tape** in place of actual device numbers must be changed before use. See "DASD and Tape Device Support" on page 3 for valid device number notation.

IEBDG EXAMPLE 1

In this example, binary zeros are placed in two fields of 100 records copied from a sequential data set. After the operation, each record in the copied data set (OUTSET) contains binary zeros in locations 20 through 29 and 50 through 59.

```
//CLEAROUT JOB  ,,MSGLEVEL=1
//         EXEC PGM=IEBDG
//SYSPRINT DD   SYSOUT=A
//SEQIN    DD   DSNAME=INSET,UNIT=tape,DISP=(OLD,KEEP),
//              DCB=(RECFM=FB,LRECL=80,BLKSIZE=800),
//              LABEL=(,NL),
//              VOLUME=SER=222222
//SEQOUT   DD   DSNAME=OUTSET,UNIT=tape,VOLUME=SER=222333,
//              DCB=(RECFM=FB,LRECL=80,BLKSIZE=800),
//              DISP=(,KEEP),
//              LABEL=(,NL)
//SYSIN    DD   *
       DSD      OUTPUT=(SEQOUT),INPUT=(SEQIN)
       FD       NAME=FIELD1,LENGTH=10,STARTLOC=20
       FD       NAME=FIELD2,LENGTH=10,STARTLOC=50
       CREATE   QUANTITY=100,INPUT=SEQIN,NAME=(FIELD1,FIELD2)
       END
/*
```

The control statements are discussed below:

- SEQIN DD defines a sequential input data set (INSET). The data set was originally written on a unlabeled tape volume.
- SEQOUT DD defines the test data set (OUTSET). The output records are identical to the input records, except for locations 20 through 29 and 50 through 59, which contain binary zeros at the completion of the operation.
- SYSIN DD defines the control data set, which follows in the input stream.
- DSD marks the beginning of a set of utility control statements and refers to the DD statements defining the input and output data sets.
- The first FD statement defines an 80-byte field of input data.
- The first and second FD statements create two 10-byte fields (FIELD1 and FIELD2) that contain binary zeros. The fields are to begin in the 20th and 50th bytes of each output record.
- CREATE constructs 100 output records in which the contents of previously defined fields (FIELD1, FIELD2) are placed in their respective starting locations in each of the output records. Input records from data set INSET are used as the basis of the output records.
- END signals the end of a set of utility control statements.

IEBDG EXAMPLE 2

In this example, a 10-byte alphabetic pattern is rippled. At the end of the job step the first output record contains "ABCDEFGHIJ," followed by data in location 11 through 80 from the input record; the second record contains "BCDEFGHIJK" followed by data in locations 11 through 80, etc.

```
                                                                      72
//RIPPLE   JOB  ,,MSGLEVEL=1
//         EXEC PGM=IEBDG
//SYSPRINT DD   SYSOUT=A
//SEQIN    DD   DSNAME=INSET,DISP=(OLD,KEEP),VOL=SER=222222,
//              DCB=(RECFM=FB,LRECL=80,BLKSIZE=800),
//              UNIT=tape
//SEQOUT   DD   DSNAME=OUTSET,UNIT=disk,VOLUME=SER=111111,
//              DCB=(RECFM=FB,LRECL=80,BLKSIZE=800),
//              DISP=(,KEEP),
//              SPACE=(TRK,(10,10))
//SYSIN    DD   *
       DSD      OUTPUT=(SEQOUT),INPUT=(SEQIN)
       FD       NAME=FIELD1,INPUT=SEQIN,LENGTH=80
       FD       NAME=FIELD2,LENGTH=10,FORMAT=AL,ACTION=RP,  C
                STARTLOC=1
       CREATE   QUANTITY=100,INPUT=SEQIN,NAME=(FIELD1,FIELD2)
       END
/*
```

The control statements are discussed below:

- SEQIN DD defines an input sequential data set (INSET). The data set was originally written on a 9-track, standard labeled tape volume.
- SEQOUT DD defines the test output data set (OUTSET). Ten tracks of primary space and ten tracks of secondary space are allocated for the sequential data set on a disk volume.
- SYSIN DD defines the control data set, which follows in the input stream.
- DSD marks the beginning of a set of utility control statements and refers to the DD statements defining the input and output data sets.
- The FD statements create a 10-byte field in which the pattern ABCDEFGHIJ is initially placed. The data is rippled after each output record is written.
- CREATE constructs 100 output records in which the contents of a previously defined field (FIELD1) are included. The CREATE statement uses input records from data set INSET as the basis of the output records.
- END signals the end of a set of utility control statements.

IEBDG EXAMPLE 3

In this example, output records are created entirely from utility control statements. Three fields are created and used in the construction of the output records. In two of the fields, alphabetic data is truncated; the other field is a numeric field that is incremented (indexed) by one after each output record is written. Figure 47 on page 124 shows the contents of the output records at the end of the job step.

Field 1	Field 2	Field 3 (packed decimal)	
1	31	61 71	80
ABCDEFGHIJKLMNOPQRSTUVWXYZABCD	ABCDEFGHIJKLMNOPQRSTUVWXYZABCD	FF . . . FF	123 . . . 90
BCDEFGHIJKLMNOPQRSTUVWXYZABCD	ABCDEFGHIJKLMNOPQRSTUVWXYZABC	FF . . . FF	123 . . . 91
CDEFGHIJKLMNOPQRSTUVWXYZABCD	ABCDEFGHIJKLMNOPQRSTUVWXYZAB	FF . . . FF	123 . . . 92
DEFGHIJKLMNOPQRSTUVWXYZABCD	ABCDEFGHIJKLMNOPQRSTUVWXYZA	FF . . . FF	123 . . . 93
EFGHIJKLMNOPQRSTUVWXYZABCD	ABCDEFGHIJKLMNOPQRSTUVWXYZ	FF . . . FF	123 . . . 94

Figure 47. Output Records at Job Step Completion

```
                                                                      72
//UTLYONLY JOB  ,,MSGLEVEL=1
//         EXEC PGM=IEBDG
//SYSPRINT DD   SYSOUT=A
//SEQOUT   DD   DSNAME=OUTSET,UNIT=disk,DISP=(,KEEP),
//              DCB=(RECFM=FB,LRECL=80,BLKSIZE=800),
//              SPACE=(TRK,(10,10)),
//              VOLUME=SER=111111
//SYSIN    DD   DATA
   DSD OUTPUT=(SEQOUT)
   FD  NAME=FIELD1,LENGTH=30,STARTLOC=1,FORMAT=AL,ACTION=TL
   FD  NAME=FIELD2,LENGTH=30,STARTLOC=31,FORMAT=AL,ACTION=TR
   FD  NAME=FIELD3,LENGTH=10,STARTLOC=71,PICTURE=10,          C
         P'1234567890',INDEX=1
   CREATE QUANTITY=100,NAME=(FIELD1,FIELD2,FIELD3),FILL=X'FF'
   END
/*
```

The control statements are discussed below:

- SEQOUT DD defines the test output data set. Ten tracks of primary space and ten tracks of secondary space are allocated for the sequential data set on a disk volume.
- SYSIN DD defines the control data set, which follows in the input stream.
- DSD marks the beginning of a set of utility control statements and refers to the DD statement defining the output data set.
- FD defines the contents of three fields to be used in the construction of output records. The first field contains 30 bytes of alphabetic data to be truncated left after each output record is written. The second field contains 30 bytes of alphabetic data to be truncated right after each output record is written. The third field is a 10-byte field containing a packed decimal number (1234567890) to be increased by one after each record is written.
- CREATE constructs 100 output records in which the contents of previously defined fields (FIELD1, FIELD2, and FIELD3) are included. Note that after each record is written, FIELD1 and FIELD2 are restored to full width.
- END signals the end of a set of utility control statements.

IEBDG EXAMPLE 4

In this example, two partitioned members and input records from the input stream are used as the basis of a partitioned output member. Each block of 12 output records contains 10 modified records from an input partitioned member and two records from the input stream. Figure 48 shows the content of the output partitioned member at the end of the job step.

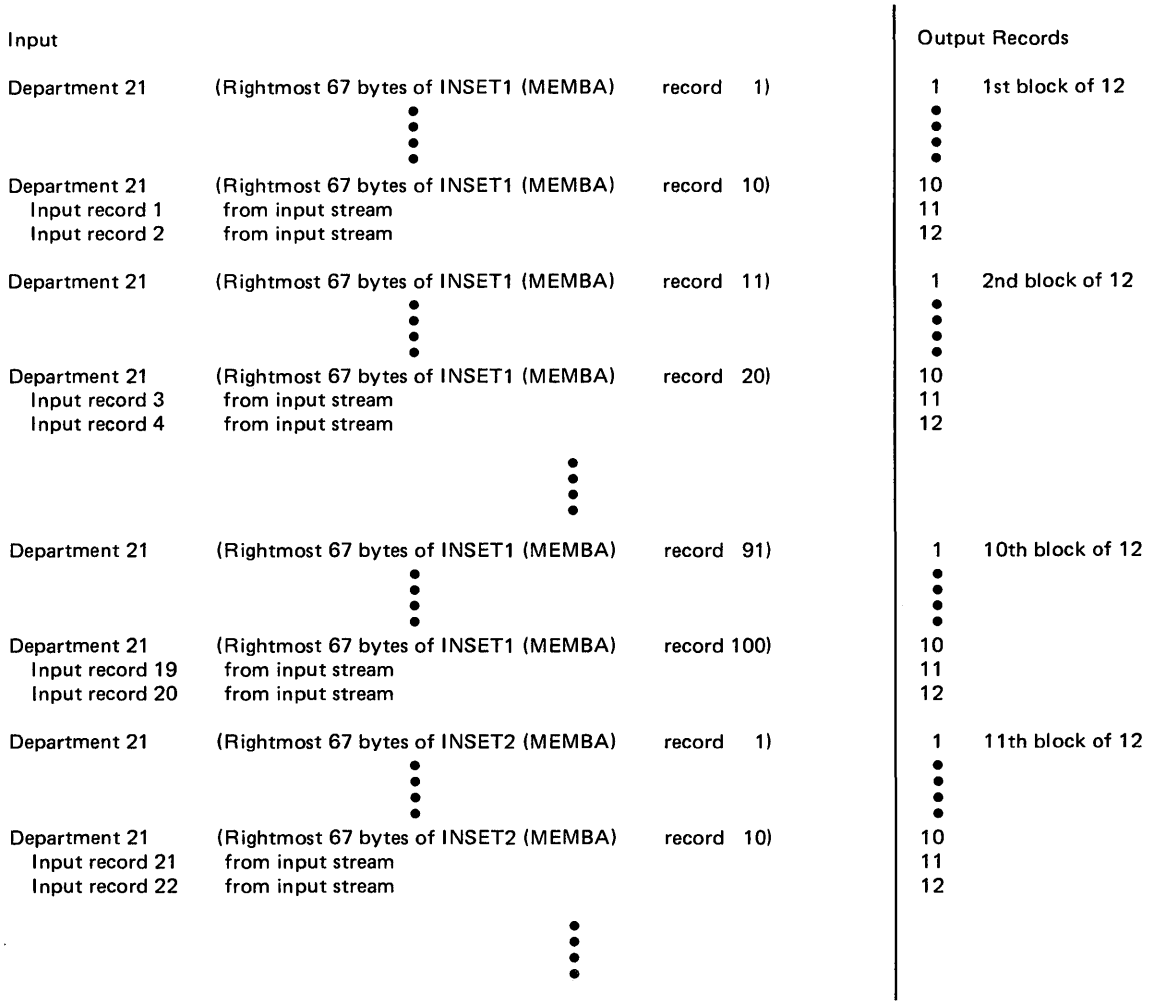


Figure 48. Output Partitioned Member at Job Step Completion

```
//MIX      JOB  ,,MSGLEVEL=1
//         EXEC PGM=IEBDG
//SYSPRINT DD   SYSOUT=A
//PARIN1   DD   DSNAME=INSET1(MEMBA),UNIT=disk,DISP=OLD,
//              DCB=(RECFM=FB,LRECL=80,BLKSIZE=800,DSORG=PS),
//              VOLUME=SER=111111
//PARIN2   DD   DSNAME=INSET2(MEMBA),UNIT=disk,DISP=OLD,
//              DCB=(RECFM=FB,LRECL=80,BLKSIZE=960,DSORG=PS),
//              VOLUME=SER=222222
//PAROUT   DD   DSNAME=PARSET(MEMBA),UNIT=disk,DISP=(,KEEP),
//              VOLUME=SER=333333,SPACE=(TRK,(10,10,5)),
//              DCB=(RECFM=FB,LRECL=80,BLKSIZE=960,DSORG=PS)
//SYSIN    DD   DATA
   DSD     OUTPUT=(PAROUT),INPUT=(PARIN1,PARIN2)
   FD      NAME=FIELD1,LENGTH=13,PICTURE=13,'DEPARTMENT 21'
   REPEAT  QUANTITY=10,CREATE=2
   CREATE  QUANTITY=10,INPUT=PARIN1,NAME=FIELD1
   CREATE  QUANTITY=2,INPUT=SYSIN

(input records 1 through 20)

   REPEAT  QUANTITY=10,CREATE=2
   CREATE  QUANTITY=10,INPUT=PARIN2,NAME=FIELD1
   CREATE  QUANTITY=2,INPUT=SYSIN

(input records 21 through 40)

   END
/*
```

The control statements are discussed below:

- PARIN1 DD defines one of the input partitioned members.
- PARIN 2 DD defines the second of the input partitioned members. (Note that the members are from different partitioned data sets.)
- PAROUT DD defines the output partitioned member. This example assumes that the partitioned data set does not exist prior to the job step; that is, this DD statement allocates space for the partitioned data set.
- SYSIN DD defines the control data set, which follows in the input stream.
- DSD marks the beginning of a set of utility control statements and refers to the DD statements defining the input and output data sets.
- FD creates a 13-byte field in which the picture "DEPARTMENT 21" is placed.
- The first REPEAT statement indicates that the following group of two CREATE statements is to be repeated 10 times.
- The first CREATE statement creates 10 output records. Each output record is constructed from an input record (from partitioned data set INSET1) and from previously defined FIELD1.
- The second CREATE statement indicates that two records are to be constructed from input records included next in the input stream.

- The E record separates the input records from the REPEAT statement. The next REPEAT statement group is identical to the preceding group, except that records from a different partitioned member are used as input.
- END signals the end of a set of utility control statements.

IEBDG EXAMPLE 5

In this example, output records are created from three sets of utility control statements and written in three partitioned data set members. Four fields are created and used in the construction of the output records. In two of the fields (FIELD1 and FIELD3), alphabetic data is shifted. FIELD2 is fixed zoned decimal and FIELD4 is fixed alphameric. Figure 49 shows the partitioned data set members at the end of the job step.

MEMBA

Field 1 1	Field 3 31	Field 2 51	Binary zeros 71 80
ABCDEFGHIJKLMNOPQRSTUVWXYZABCD	ABCDEFGHIJKLMNOPQRST	00000000000000000001	fill
BCDEFGHIJKLMNOPQRSTUVWXYZABCD	ABCDEFGHIJKLMNOPQRS	00000000000000000001	fill
CDEFGHIJKLMNOPQRSTUVWXYZABCD	ABCDEFGHIJKLMNOPQR	00000000000000000001	fill
DEFGHIJKLMNOPQRSTUVWXYZABCD	ABCDEFGHIJKLMNOPQ	00000000000000000001	fill

MEMBB

Field 3 1	Field 3 21	Field 3 41	Field 2 61 80
ABCDEFGHIJKLMNOPQRST	ABCDEFGHIJKLMNOPQRST	ABCDEFGHIJKLMNOPQRST	00000000000000000001
ABCDEFGHIJKLMNOPQRS	ABCDEFGHIJKLMNOPQRS	ABCDEFGHIJKLMNOPQRS	00000000000000000001
ABCDEFGHIJLKMNOPQR	ABCDEFGHIJKLMNOPQR	ABCDEFGHIJKLMNOPQR	00000000000000000001
ABCDEFGHIJKLMNOPQ	ABCDEFGHIJKLMNOPQ	ABCDEFGHIJKLMNOPQ	00000000000000000001

MEMBC

Field 4 1	Field 1 31	Binary zeros 61 80
ABCDEFGHIJKLMNOPQRSTUVWXYZ0123	ABCDEFGHIJKLMNOPQRSTUVWXYZABCD	fill
ABCDEFGHIJKLMNOPQRSTUVWXYZ0123	BCDEFGHIJKLMNOPQRSTUVWXYZABCD	fill
ABCDEFGHIJKLMNOPQRSTUVWXYZ0123	CDEFGHIJKLMNOPQRSTUVWXYZABCD	fill
ABCDEFGHIJKLMNOPQRSTUVWXYZ0123	DEFGHIJKLMNOPQRSTUVWXYZABCD	fill

Figure 49. Partitioned Data Set Members at Job Step Completion

The control statements are discussed below:

- PAROUT1 DD defines the first member (MEMBA) of the partitioned output data set. This example assumes that the partitioned data set does not exist prior to this job step; that is, this DD statement allocates space for the data set.
- PAROUT2 and PAROUT3 DD define the second and third members, respectively, of the output partitioned data set. Note that each DD statement specifies DISP=OLD and UNIT=AFF=PAROUT1.
- SYSIN DD defines the control data set that follows in the input stream.
- DSD marks the beginning of a set of utility control statements and refers to the DD statement defining the member applicable to that set of utility control statements.
- FD defines the contents of a field that is used in the subsequent construction of output records.

```
//UTSTS     JOB   ,,MSGLEVEL=1
//          EXEC PGM=IEBDG
//SYSPRINT DD    SYSOUT=A
//PAROUT1  DD    DSNAME=PARSET(MEMBA),UNIT=disk,
//               DISP=(,KEEP),
//               VOLUME=SER=111111,SPACE=(TRK,(10,10,5)),
//               DCB=(RECFM=FB,LRECL=80,BLKSIZE=800,DSORG=PS)
//PAROUT2  DD    DSNAME=PARSET(MEMBB),UNIT=AFF=PAROUT1,
//               DCB=(RECFM=FB,LRECL=80,BLKSIZE=800,DSORG=PS),
//               DISP=OLD,
//               VOLUME=SER=111111
//PAROUT3  DD    DSNAME=PARSET(MEMBC),UNIT=AFF=PAROUT1,
//               DCB=(RECFM=FB,LRECL=80,BLKSIZE=800,DSORG=PS),
//               DISP=OLD,
//               VOLUME=SER=111111
//SYSIN    DD    DATA
        DSD      OUTPUT=(PAROUT1)
        FD       NAME=FIELD1,LENGTH=30,FORMAT=AL,ACTION=SL
        FD       NAME=FIELD2,LENGTH=20,FORMAT=ZD
        FD       NAME=FIELD3,LENGTH=20,FORMAT=AL,ACTION=SR
        FD       NAME=FIELD4,LENGTH=30,FORMAT=AN
        CREATE   QUANTITY=4,NAME=(FIELD1,FIELD3,FIELD2)
        END
        DSD      OUTPUT=(PAROUT2)
        CREATE   QUANTITY=4,NAME=(FIELD2,(COPY=3,FIELD3))
        END
        DSD      OUTPUT=(PAROUT3)
        CREATE   QUANTITY=4,NAME=(FIELD4,FIELD1)
        END
/*
```

- CREATE constructs four records from combinations of previously defined fields.
- END signals the end of a set of utility control statements.

IEBDG EXAMPLE 6

In this example, 10 fields containing user-supplied EBCDIC pictures are used in the construction of output records. After a record is written, each field is rolled or waved, as specified in the applicable FD statement. Figure 50 shows the contents of the output records at the end of the job step.

FIELD1	FIELD 2	FIELD3	FIELD4	FIELD5	FIELD6	FIELD7	FIELD8	FIELD9	FIELD10
AAAAA	BBBBB	A AA	BB B	AAA	CCCCC	DDDD	C CC	DD D	CCC
AAAAA	BBBBB	A AA	BB B	AAA	CCCCC	DDDD	C CC	DD D	CCC
AAAAA	BBBBB	A AA	BB B	AAA	CCCCC	DDDD	C CC	DD D	CCC
AAAAA	BBBBB	A AA	BB B	AAA	CCCCC	DDDD	C CC	DD D	CCC
AAAAA	BBBBB	A AA	BB B	AAA	CCCCC	DDDD	C CC	DD D	CCC
AAAAA	BBBBB	A AA	BB B	AAA	CCCCC	DDDD	C CC	DD D	CCC
AAAAA	BBBBB	A AA	BB B	AAA	CCCCC	DDDD	C CC	DD D	CCC
AAAAA	BBBBB	A AA	BB B	AAA	CCCCC	DDDD	C CC	DD D	CCC
AAAAA	BBBBB	A AA	BB B	AAA	CCCCC	DDDD	C CC	DD D	CCC
AAAAA	BBBBB	A AA	BB B	AAA	CCCCC	DDDD	C CC	DD D	CCC

Figure 50. Contents of Output Records at Job Step Completion

```
                                                                      72
//ROLLWAVE JOB  ,,MSGLEVEL=1
//         EXEC PGM=IEBDG
//SYSPRINT DD   SYSOUT=A
//OUTSET   DD   DSNAME=SEQSET,UNIT=disk,DISP=(,KEEP),
//              VOLUME=SER=SAMP,SPACE=(TRK,(10,10)),
//              DCB=(RECFM=FB,LRECL=80,BLKSIZE=800)
//SYSIN    DD   *
 DSD    OUTPUT=(OUTSET)
 FD     NAME=FIELD1,LENGTH=8,PICTURE=8,'   AAAAA',ACTION=RO
 FD     NAME=FIELD2,LENGTH=8,PICTURE=8,'BBBBB   ',ACTION=RO
 FD     NAME=FIELD3,LENGTH=8,PICTURE=8,'A   AA  ',ACTION=RO
 FD     NAME=FIELD4,LENGTH=8,PICTURE=8,' BB    B',ACTION=RO
 FD     NAME=FIELD5,LENGTH=8,PICTURE=8,'   AAA  ',ACTION=RO
 FD     NAME=FIELD6,LENGTH=8,PICTURE=8,'   CCCCC',ACTION=WV
 FD     NAME=FIELD7,LENGTH=8,PICTURE=8,'  DDDD  ',ACTION=WV
 FD     NAME=FIELD8,LENGTH=8,PICTURE=8,'  C  CC ',ACTION=WV
 FD     NAME=FIELD9,LENGTH=8,PICTURE=8,'  DD   D',ACTION=WV
 FD     NAME=FIELD10,LENGTH=8,PICTURE=8,'  CCC   ',ACTION=WV
 CREATE QUANTITY=300,NAME=(FIELD1,FIELD2,FIELD3,                C
               FIELD4,FIELD5,FIELD6,FIELD7,FIELD8,              C
               FIELD9,FIELD10)
 END
/*
```

The control statements are discussed below:

- OUTSET DD defines the output sequential data set on a disk volume. Ten tracks of primary space and 10 tracks of secondary space are allocated to the data set.
- SYSIN DD defines the control data set that follows in the input stream.
- DSD marks the beginning of a set of utility control statements and refers to the DD statement defining the output data set.
- FD defines a field to be used in the subsequent construction of output records. The direction and frequency of the initial roll or wave depends on the location of data in the field.
- CREATE constructs 300 records from the contents of the previously defined fields.
- END signals the end of a set of utility control statements.

IEBDG EXAMPLE 7

In this example, the first 10 bytes of the output record contain data generated in zoned decimal format. This field serves as the key field for the output record in the output indexed sequential data set. The key field is increased (indexed) by one for each record. The input sequential data set provides an additional 80-byte field to complete the output record.

```
                                                                      72
//CREATEIS JOB  MSGLEVEL=1
//BEGIN    EXEC PGM=IEBDG
//TAPEIN   DD   DCB=(BLKSIZE=80,LRECL=80,RECFM=F),
//              DISP=(OLD,KEEP),UNIT=disk,
//              LABEL=(,SL),
//              DSNAME=TAPEIT,VOL=SER=MASTER
//DISKOUT  DD   DCB=(BLKSIZE=270,LRECL=90,RECFM=FB,DSORG=IS,
//               NTM=2,OPTCD=MY,RKP=0,KEYLEN=10,CYLOFL=1),
//              UNIT=disk,SPACE=(CYL,1),
//              DISP=(NEW,KEEP),
//              VOL=SER=111111,DSNAME=CREATIS
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
  DSD    OUTPUT=(DISKOUT),INPUT=(TAPEIN)
  FD     NAME=DATAFD,LENGTH=80,FROMLOC=1,                            C
         STARTLOC=11,INPUT=TAPEIN
  FD     NAME=KEYFD,LENGTH=10,STARTLOC=1,FORMAT=ZD,INDEX=1
  CREATE INPUT=TAPEIN,NAME=(KEYFD,DATAFD)
  END
/*
```

The control statements are discussed below:

- TAPEIN DD defines the sequential input data set.
- DISKOUT DD defines the indexed sequential output data set.
- SYSIN DD defines the control data set, which follows in the input stream.
- DSD marks the beginning of a set of utility control statements and refers to the DD statement defining the output data set.
- FD defines a field that will be used in the subsequent construction of output records. The first FD statement in this example defines and locates an 80-byte field of input data. The data is field selected from one of the input logical records and placed at start location 11 of the output logical record. The second FD statement defines and locates the 10-byte key field.
- CREATE constructs a 90-byte output record by referring to the previously defined fields.
- END signals the end of a set of utility control statements.

IEBEDIT PROGRAM

IEBEDIT is a data set utility used to create an output data set containing a selection of jobs or job steps. At a later time, data sets defined on tape volumes and direct access devices can be used as input streams for job processing.

IEBEDIT creates an output job stream by editing and selectively copying a job stream provided as input. The program can copy:

- An entire job or jobs, including JOB statements and any associated JOBLIB or JOBCAT statements, and JES2 or JES3 control statements.
- Selected job steps, including the JOB statement, JES2 or JES3 control statements following the JOB statement, and any associated JOBLIB or JOBCAT statements.

All selected JOB statements, JES2 or JES3 control statements, JOBLIB or JOBCAT statements, jobs, or job steps are placed in the output data set in the same order as they exist in the input data set. A JES2 or JES3 control statement or a JOBLIB or JOBCAT statement is copied only if it follows a selected JOB statement.

When IEBEDIT encounters a selected job step containing an input record having the characters "..*" (period, period, asterisk) in columns 1 through 3, the program automatically converts that record to a termination statement (/* statement) and places it in the output data set.

A "/*nonblank" indicates a JES2 or JES3 control statement.

INPUT AND OUTPUT

IEBEDIT uses the following input:

- An input data set, which is a sequential data set consisting of a job stream. The input data set is used as source data in creating an output sequential data set.
- A control data set, which contains utility control statements that are used to specify the organization of jobs and job steps in the output data set.

IEBEDIT produces the following output:

- An output data set, which is a sequential data set consisting of a resultant job stream.
- A message data set, which is a sequential data set that contains applicable control statements, error messages, if applicable, and, optionally, the output data set.

RETURN CODES

IEBEDIT returns a code in register 15 to indicate the results of program execution. The return codes and their meanings are listed below.

Codes	Meaning
00 (00 hex)	Successful completion.
04 (04)	An error occurred. The output data set may not be usable as a job stream. Processing continues.
08 (08)	An unrecoverable error occurred while attempting to process the input, output, or control data set. The job step is terminated.

Figure 51. IEBEDIT Return Codes

CONTROL

IEBEDIT is controlled by job control statements and utility control statements. The job control statements are required to execute or invoke the program and to define the data sets used and produced by the program. The utility control statements are used to control the functions of the program.

JOB CONTROL STATEMENTS

Figure 52 shows the job control statements for IEBEDIT.

Statement	Use
JOB	Initiates the job.
EXEC	Specifies the program name (PGM=IEBEDIT) or, if the job control statements reside in a procedure library, the procedure name.
SYSPRINT DD	Defines a sequential message data set. The data set can be written to a system output device, a tape volume, or a direct access volume.
SYSUT1 DD	Defines a sequential input data set on a card reader, tape volume, or direct access device.
SYSUT2 DD	Defines a sequential output data set on a card punch, printer, tape volume, or direct access device.
SYSIN DD	Defines the control data set. The data set normally is included in the input stream; however, it can be defined as a member of a procedure library or as a sequential data set existing somewhere other than in the input stream.

Figure 52. Job Control Statements for IEBEDIT

Notes to Figure 52:

1. The block size for the SYSPRINT data set must be a multiple of 121. If not, the job step is terminated with a return code of 8. The block size for the SYSIN, SYSUT1, and SYSUT2 data sets must be a multiple of 80. Any blocking factor can be specified for these record sizes.

2. Any JES2 or JES3 control statement or JOBLIB DD statement that follows a selected JOB statement is automatically copied to the output data set.

   JES2 or JES3 control statements preceding the JOB statement are assumed to belong to the previous job. JES2 or JES3 control statements preceding the first JOB statement are included only if a total copy is requested.

   JES2 or JES3 control statements within a DD DATA stream are included only if a delimiter other than "/*" is coded in the DD DATA card. For a description of coding another delimiter, see the publication JCL. If another delimiter is not coded, the first two characters of the JES2 or JES3 control statement will act as a delimiter to DD DATA.

UTILITY CONTROL STATEMENT

IEBEDIT uses only one utility control statement, EDIT. Continuation requirements for the statement are described in "Continuing Utility Control Statements" on page 5.

EDIT Statement

The EDIT statement indicates which step or steps of a specified job in the input data set are to be included in the output data set. Any number of EDIT statements can be included in an operation, thus including selected jobs in the output data set.

EDIT statements must be included in the same order as the input jobs that they represent. If no EDIT statement is present in the control data set, the entire input data set is copied.

The format of the EDIT statement is:

[label]	**EDIT**	**[START=**jobname**]** **[,TYPE=**POSITION**|INCLUDE|EXCLUDE]** **[,STEPNAME=(**name**[,**name-name**]),...]** **[,NOPRINT]**

Parameters	Applicable Control Statements	Description of Parameters
NOPRINT	EDIT	**NOPRINT** specifies that the message data set is not to include a listing of the output data set. **Default:** The resultant output is listed in the message data set.

Parameters	Applicable Control Statements	Description of Parameters
START	EDIT	**START=**jobname specifies the name of the input job to which the EDIT statement applies. (Each EDIT statement must apply to a separate job.) If START is specified without TYPE and STEPNAME, the JOB statement and all job steps for the specified job are included in the output. **Default:** If START is omitted and only one EDIT statement is provided, the first job encountered in the input data set is processed. If START is omitted from an EDIT statement other than the first statement, processing continues with the next JOB statement found in the input data set.
STEPNAME	EDIT	**STEPNAME=(**name**[,**name-name**]),...** specifies the first job step to be placed in the output data set when coded with TYPE=POSITION. Job steps preceding this step are not copied to the output data set. name can be specified as a single job step name or a sequential range of names, separated by a hyphen: name-name. If more than one value is specified for name, the entire STEPNAME field must be enclosed in parentheses. When coded with TYPE=INCLUDE or TYPE=EXCLUDE, STEPNAME specifies the names of job steps that are to be included in or excluded from the operation. For example, STEPNAME=(STEPA,STEPF-STEPL,STEPZ) indicates that job steps STEPA, STEPF through STEPL, and STEPZ are to be included in or excluded from the operation. **Default:** If STEPNAME is omitted, the entire input job whose name is specified on the EDIT statement is copied. If no job name is specified, the first job encountered is processed.

Parameters	Applicable Control Statements	Description of Parameters
TYPE	EDIT	**TYPE=POSITION|INCLUDE|EXCLUDE** specifies the contents of the output data set. These values can be coded: **POSITION** specifies that the output is to consist of a JOB statement, the job step specified in the STEPNAME parameter, and all steps that follow it. All job steps preceding the specified step are omitted from the operation. POSITION is the default. **INCLUDE** specifies that the output data set is to contain a JOB statement and all job steps specified in the STEPNAME parameter. **EXCLUDE** specifies that the output data set is to contain a JOB statement and all job steps belonging to the job except those steps specified in the STEPNAME parameter.

IEBEDIT EXAMPLES

The following examples show some of the uses of IEBEDIT. Figure 53 can be used as a quick-reference guide to IEBEDIT examples. The numbers in the "Example" column refer to examples that follow.

Operation	Devices	Comments	Example
COPY	9-track Tape	The input data set contains three jobs. One job is to be copied.	1
COPY	7-track Tape	The output data set is the second data set on the volume. One job step is to be copied from each of three jobs.	2
COPY	Disk and 9-track Tape	Include a job step from one job and exclude a job step from another job.	3
COPY	Disk	Latter portion of a job stream is to be copied.	4

Figure 53 (Part 1 of 2). IEBEDIT Example Directory

Operation	Devices	Comments	Example
COPY	9-track Tape	All records in the input data set are to be copied. The "..*" record is converted to a "/*" statement in the output data set.	5
COPY	9-track Tape	The input contains a JES2 or JES3 control statement and a new delimiter.	6

Figure 53 (Part 2 of 2). IEBEDIT Example Directory

Examples that use **disk** or **tape** in place of actual device numbers must be changed before use. See "DASD and Tape Device Support" on page 3 for valid device number notation.

IEBEDIT EXAMPLE 1

In this example, one job (JOBA), including all of its job steps (A, B, C, and D), is copied into the output data set. The input data set contains three jobs: JOBA, which has four job steps; JOBB, which has three job steps; and JOBC, which has two job steps.

```
//EDIT1    JOB  09#440,SMITH
//         EXEC PGM=IEBEDIT
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   UNIT=tape,DISP=(OLD,KEEP),VOL=SER=001234
//SYSUT2   DD   UNIT=tape,DISP=(NEW,KEEP),VOL=SER=001235,
//              DCB=(RECFM=F,LRECL=80,BLKSIZE=80),
//              DSNAME=OUTTAPE
//SYSIN    DD   *
         EDIT  START=JOBA
/*
```

The control statements are discussed below:

- SYSUT1 DD defines the input data set. The data set resides on a standard labeled tape volume (001234).
- SYSUT2 DD defines the output data set, called OUTTAPE. The data set is to reside as the first data set on a standard labeled tape volume (001235).
- SYSIN DD defines the control data set, which follows in the input stream.
- EDIT indicates that JOBA is to be copied in its entirety.

IEBEDIT EXAMPLE 2

This example copies one job step from each of three jobs. The input data set contains three jobs: JOBA, which includes STEPA, STEPB, STEPC, and STEPD; JOBB, which includes STEPE, STEPF, and STEPG; and JOBC, which includes STEPH and STEPJ.

```
//EDIT2    JOB  09#440,SMITH
//         EXEC PGM=IEBEDIT
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DISP=(OLD,KEEP),VOLUME=SER=001234,
//              UNIT=tape
//SYSUT2   DD   DSN=OUTSTRM,UNIT=tape,DISP=(NEW,KEEP),
//              DCB=(RECFM=F,LRECL=80,BLKSIZE=80),
//              LABEL=(2,SL)
//SYSIN    DD   *
       EDIT   START=JOBA,TYPE=INCLUDE,STEPNAME=(STEPC,STEPD)
       EDIT   START=JOBB,TYPE=INCLUDE,STEPNAME=STEPE
       EDIT   START=JOBC,TYPE=INCLUDE,STEPNAME=STEPJ
/*
```

The control statements are discussed below:

- SYSUT1 DD defines the input data set. The data set resides on a standard labeled tape volume (001234).
- SYSUT2 DD defines the output data set, OUTSTRM. The data set is to reside as the second data set on a standard labeled tape volume (001235).
- SYSIN DD defines the control data set, which follows in the input stream.
- The EDIT statements copy the JOB statements and job steps described as follows:

  1. The JOB statement and steps STEPC and STEPD for JOBA.
  2. The JOB statement and STEPE for JOBB.
  3. The JOB statement and STEPJ for JOBC.

IEBEDIT EXAMPLE 3

This example includes a job step from one job and excludes a job step from another job. The input data set contains three jobs: JOBA, which includes STEPA, STEPB, STEPC, and STEPD; JOBB, which includes STEPE, STEPF, and STEPG; and JOBC, which includes STEPH and STEPJ.

```
//EDIT3    JOB  09#440,SMITH
//         EXEC PGM=IEBEDIT
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=INSET,UNIT=disk,DISP=(OLD,KEEP),
//              VOLUME=SER=111111
//SYSUT2   DD   DSNAME=OUTTAPE,UNIT=tape,LABEL(,NL),
//              DCB=(DEN=2,RECFM=F,LRECL=80,BLKSIZE=80),
//              DISP=(,KEEP)
//SYSIN    DD   *
       EDIT   START=JOBB,TYPE=INCLUDE,STEPNAME=(STEPF-STEPG)
       EDIT   START=JOBC,TYPE=EXCLUDE,STEPNAME=STEPJ
/*
```

The control statements are discussed below:

- SYSUT1 DD defines the input data set, INSET. The data set resides on a disk volume (111111).

- SYSUT2 DD defines the output data set, OUTTAPE. The data set is to reside as the first or only data set on an unlabeled (800 bits per inch) tape volume.
- SYSIN DD defines the control data set, which follows in the input stream.
- The EDIT statements copy JOB statements and job steps as described below:

  1. The JOB statement and steps STEPF and STEPG for JOBB.
  2. The JOB statement and STEPH, excluding STEPJ, for JOBC.

IEBEDIT EXAMPLE 4

This example copies the JOBA JOB statement, the job step STEPF, and all the steps that follow it. The input data set contains one job (JOBA), which includes STEPA, STEPB, . . . STEPL. Job steps STEPA through STEPE are not included in the output data set.

```
//EDIT4    JOB  09#440,SMITH
//         EXEC PGM=IEBEDIT
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=INSTREAM,UNIT=disk,
//              DISP=(OLD,KEEP),
//              VOLUME=SER=111111
//SYSUT2   DD   DSNAME=OUTSTREM,UNIT=disk,
//              DISP=(,KEEP),
//              DCB=(RECFM=F,LRECL=80,BLKSIZE=80),
//              VOLUME=SER=222222,
//              SPACE=(TRK,2)
//SYSIN    DD   *
        EDIT   START=JOBA,TYPE=POSITION,STEPNAME=STEPF
/*
```

The control statements are discussed below:

- SYSUT1 DD defines the input data set, called INSTREAM. The data set resides on a disk volume (111111).
- SYSUT2 DD defines the output data set, called OUTSTREAM. The data set is to reside on a disk volume (222222). Two tracks are allocated for the output data set.
- SYSIN DD defines the control data set, which follows in the input stream.
- EDIT copies the JOBA JOB statement and job steps STEPF through STEPL.

IEBEDIT EXAMPLE 5

This example copies the entire input (SYSUT1) data set. The record containing the characters "..*" in columns 1 through 3 is converted to a "/* " statement in the output data set.

```
//EDIT5    JOB  09#440,SMITH
//         EXEC PGM=IEBEDIT
//SYSPRINT DD   SYSOUT=A
//SYSUT2   DD   DSNAME=OUTTAPE,UNIT=tape,
//              VOLUME=SER=001234,
//              DCB=(RECFM=F,LRECL=80,BLKSIZE=80),
//              DISP=(NEW,KEEP)
//SYSIN    DD   DUMMY
//SYSUT1   DD   DATA
//BLDGDGIX JOB
//         EXEC PGM=IEHPROGM
//SYSPRINT DD   SYSOUT=A
//DD1      DD   UNIT=disk,VOLUME=SER=111111,DISP=OLD
//SYSIN    DD   *
         BLDG  INDEX=A.B.C,ENTRIES=10,EMPTY
..*
/*
```

The control statements are discussed below:

- SYSUT2 DD defines the output data set, called OUTTAPE. The data set is to reside as the first data set on a tape volume (001234).
- SYSIN DD defines a dummy control data set.
- SYSUT1 DD defines the input data set, which follows in the input stream. The job is terminated when the termination statement (/*b) is encountered. (SYSUT1 therefore includes the BLDGDGIX JOB statement, EXEC statement, SYSPRINT, DD1 and SYSIN DD statements.)

IEBEDIT EXAMPLE 6

This example copies the entire input (SYSUT1) data set, including the JES2 control statement, since a new delimiter (JP) has been coded. Otherwise, the "/*" the JES2 control statement would have terminated the input.

```
//EDIT6    JOB  09#440,SMITH
//STEPA    EXEC PGM=IEBEDIT
//SYSPRINT DD   SYSOUT=A
//SYSUT2   DD   DSN=TAPEOUT,UNIT=tape,
//              VOL=SER=001234,LABEL=(,SL),
//              DCB=(RECFM=FB,LRECL=80,BLKSIZE=800),
//              DISP=(NEW,KEEP)
//SYSIN    DD   DUMMY
//SYSUT1   DD   DATA,DLM=JP
//LISTVTOC JOB  09#550,BLUE
/*MESSAGE  JOB NEEDS VOLUME 338000
//FSTEP    EXEC PGM=IEHLIST
//SYSPRINT DD   SYSOUT=A
//DD2      DD   UNIT=disk,VOL=SER=111111,DISP=OLD
//SYSIN    DD   *
    LISTVTOC FORMAT,VOL=disk=111111
/*
JP
```

The control statements are discussed below:

- SYSUT2 DD defines the output data set, called TAPEOUT. The data set will be the first data set on a standard label tape volume (001234).
- SYSIN DD defines a dummy control data set.
- SYSUT1 DD defines the input data set, which follows in the input stream. The DLM parameter defines characters JP to act as a delimiter for the input data.
- IEBEDIT copies the JOB statement through the "/*" statement (including the LISTVTOC and MESSAGE job statements, FSTEP EXEC statement, and SYSPRINT, DD2 and SYSIN DD statements).

IEBGENER is a data set utility that can be used to:

- Create a backup copy of a sequential data set or a partitioned member.
- Produce a partitioned data set or member from a sequential input data set.
- Expand an existing partitioned data set by creating partitioned members and merging them into the data set that is to be expanded.
- Produce an edited sequential or partitioned data set.
- Reblock or change the logical record length of a data set.
- Copy user labels on sequential output data sets. (Refer to Appendix C, "Processing User Labels" on page 446.)
- Provide optional editing facilities and exits for user routines that process labels, manipulate input data, create keys, and handle permanent input/output errors. Refer to Appendix A, "Exit Routine Linkage" on page 438 for a discussion of linkage conventions that are applicable when user routines are provided.

CREATING A BACKUP COPY

A backup copy of a sequential data set or partitioned member can be produced by copying the data set or member to any IBM-supported output device. For example, a copy can be made from tape to tape, from DASD to tape, etc.

A data set that resides on a direct access volume can be copied to its own volume, provided that its data set name is changed. A partitioned data set cannot reside on a magnetic tape volume.

PRODUCING A PARTITIONED DATA SET FROM SEQUENTIAL INPUT

Through the use of utility control statements, the user can logically divide a sequential data set into **record groups** and assign member names to the record groups. IEBGENER places the newly created members in a partitioned output data set.

A partitioned data set cannot be produced if an input or output data set contains spanned records.

Figure 54 on page 142 shows how a partitioned data set is produced from a sequential data set used as input. The left side of the figure shows the sequential data set. Utility control statements are used to divide the sequential data set into record groups and to provide a member name for each record group. The right side of the figure shows the partitioned data set produced from the sequential input.

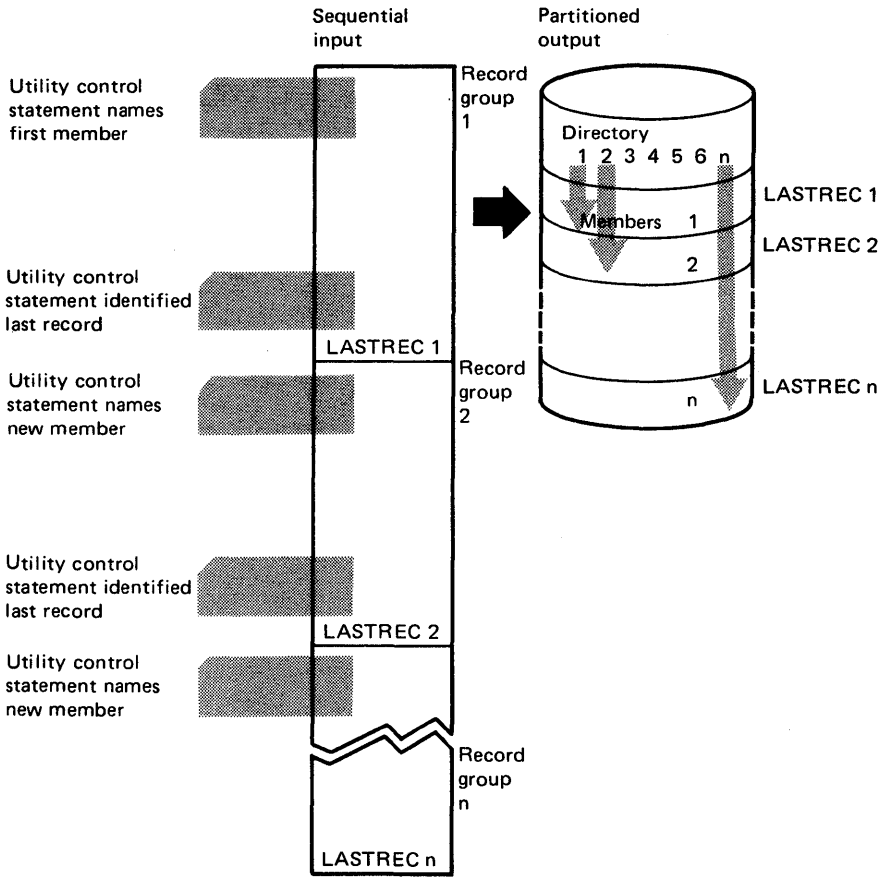


Figure 54. Creating a Partitioned Data Set from Sequential Input Using IEBGENER

EXPANDING A PARTITIONED DATA SET

An expanded data set is a data set into which an additional member or members have been merged. IEBGENER creates the members from sequential input and places them in the data set being expanded. The merge operation—the ordering of the partitioned directory—is automatically performed by the program.

Figure 55 on page 143 shows how sequential input is converted into members that are merged into an existing partitioned data set. The left side of the figure shows the sequential input that is to be merged with the partitioned data set shown in the middle of the figure. Utility control statements are used to divide the sequential data set into record groups and to provide a member name for each record group. The right side of the figure shows the expanded partitioned data set. Note that members B, D, and F from the sequential data set were placed in **available space** and that they are sequentially ordered in the partitioned directory.

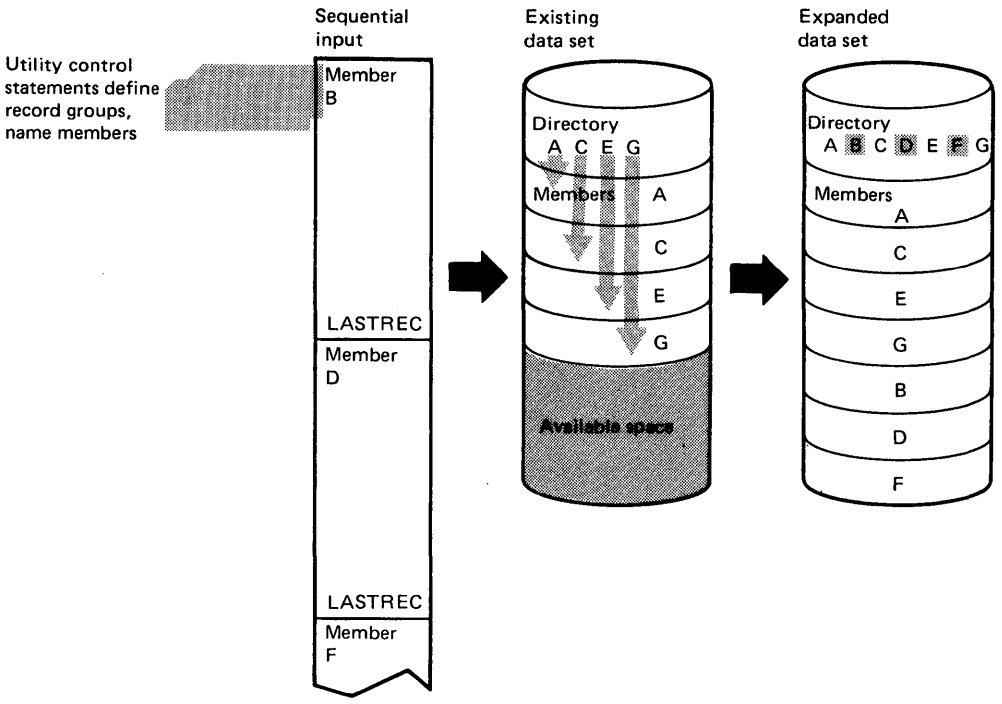


Figure 55. Expanding a Partitioned Data Set Using IEBGENER

PRODUCING AN EDITED DATA SET

IEBGENER can be used to produce an edited sequential or partitioned data set. Through the use of utility control statements, the user can specify editing information that applies to a record, a group of records, selected groups of records, or an entire data set.

An edited data set can be produced by:

- Rearranging or omitting defined data fields within a record.
- Supplying literal information as replacement data.
- Converting data from packed decimal to unpacked decimal mode, unpacked decimal to packed decimal mode, or BCD[4] to EBCDIC mode. Refer to Data Management Services for more information on converting from BCD to EBCDIC.

Figure 56 on page 144 shows part of an edited sequential data set. The left-hand side of the figure shows the data set before editing is performed. Utility control statements are used to identify the record groups to be edited and to supply editing information. In this figure, literal replacement information is supplied for information within a defined field. (Data is rearranged, omitted, or converted in the same manner.) The BBBB field in each record in the record group is to be replaced by CCCC. The right-hand side of the figure shows the data set after editing.

[4] Used here to mean the standard H character set of Binary Coded Decimal.

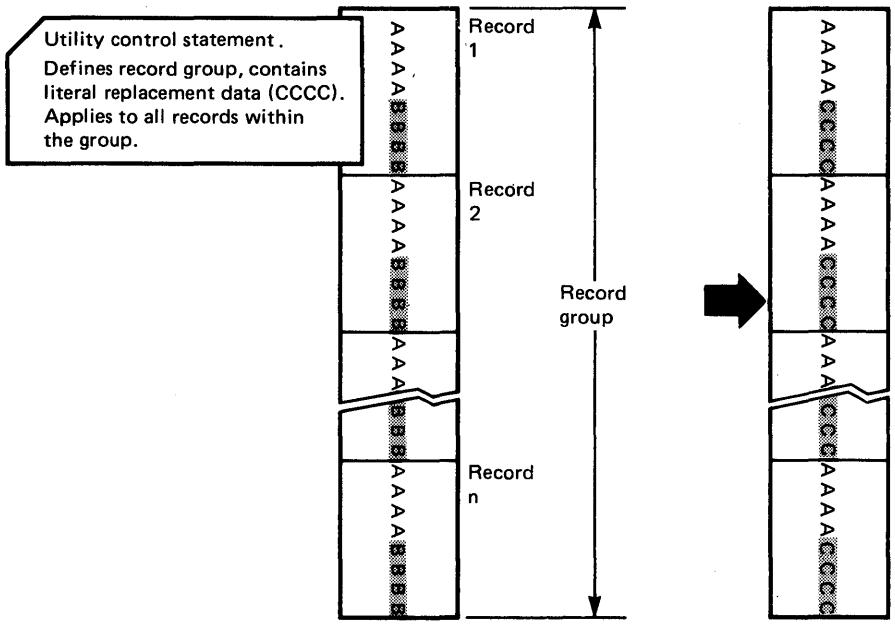


Figure 56. Editing a Sequential Data Set Using IEBGENER

IEBGENER cannot be used to edit a data set if the input and output data sets consist of variable spanned (VS) or variable blocked spanned (VBS) records and have equal block sizes and logical record lengths. In these cases, any utility control statements that specify editing are ignored. That is, for each physical record read from the input data set, the utility writes an unedited physical record on the output data set.

REBLOCKING OR CHANGING LOGICAL RECORD LENGTH

IEBGENER can be used to produce a reblocked output data set containing either fixed-length or variable-length records. In addition, the program can produce an output data set having a logical record length that differs from the input logical record length.

INPUT AND OUTPUT

IEBGENER uses the following input:

- An input data set, which contains the data that is to be copied, edited, converted into a partitioned data set, or converted into members to be merged into an existing data set. The input is either a sequential data set or a member of a partitioned data set.
- A control data set, which contains utility control statements. The control data set is required if editing is to be performed or if the output data set is to be a partitioned data set.

IEBGENER produces the following output:

- An output data set, which can be either sequential or partitioned. The output data set can be either a new data set (created during the current job step) or an existing

partitioned data set that was expanded. If a partitioned data set is created, it is a new member with a new directory entry. None of the information is copied from the previous directory entry.

- A message data set, which contains informational messages (for example, the contents of utility control statements) and any error messages.
- Message IEC507D will be issued twice when adding data or members to an existing data set which has an unexpired expiration date. This occurs because the input and output data sets are opened twice.

RETURN CODES

IEBGENER returns a code in register 15 to indicate the results of program execution. The return codes and their meanings are listed below.

Codes	Meaning
00 (00 hex)	Successful completion.
04 (04)	Probable successful completion. A warning message is written.
08 (08)	Processing was terminated after the user requested processing of user header labels only.
12 (0C)	An unrecoverable error exists. The job step is terminated.
16 (10)	A user routine passed a return code of 16 to IEBGENER. The job step is terminated.

Figure 57. IEBGENER Return Codes

CONTROL

IEBGENER is controlled by job control statements and utility control statements. The job control statements execute or invoke IEBGENER and define the data sets that are used and produced by the program. The utility control statements control the functions of IEBGENER.

JOB CONTROL STATEMENTS

Figure 58 on page 146 shows the job control statements for IEBGENER.

IEBGENER always uses two buffers, regardless of what was specified in the DCB.

SYSPRINT DD Statement

The SYSPRINT DD statement is required for each use of IEBGENER. The block size for the SYSPRINT data set must be a multiple of 121. Any blocking factor can be specified for this record size.

SYSUT1 DD Statement

The input data set for IEBGENER, as specified in SYSUT1, can contain fixed, variable, undefined, or variable spanned records.

Statement	Use
JOB	Initiates the job.
EXEC	Specifies the program name (PGM=IEBGENER) or, if the job control statements reside in a procedure library, the procedure name.
SYSPRINT DD	Defines a sequential message data set. The data set can be written to a system output device, a tape volume, or a DASD volume.
SYSUT1 DD	Defines the input data set. It can define a sequential data set or a member of a partitioned data set.
SYSUT2 DD	Defines the output data set. It can define a sequential data set, a member of a partitioned data set, or a partitioned data set.
SYSIN DD	Defines the control data set, or specifies DUMMY when the output is sequential and no editing is specified. The control data set normally resides in the input stream; however, it can be defined as a member in a partitioned data set.

Figure 58. Job Control Statements for IEBGENER

Concatenated data sets with unlike attributes are not allowed as input to IEBGENER. For information on concatenated data sets, see Data Management Services.

// cards (JCL statements) cannot be included in the SYSUT1 data set unless SYSUT1 DD DATA is specified.

Block size must be specified for the input data set in one of two ways:

- with the BLKSIZE parameter in the DD statement
- in the DCB information on the tape label.

The default RECFM is U for the input data set. RECFM must be specified if the data set is new, undefined, a dummy data set, or a data set from a card punch.

The input LRECL must be specified when the record format is FB, VS, or VBS, or when the data set is new, a dummy data set, or a data set from a card punch. In all other cases, a default LRECL is generated by IEBGENER.

A partitioned data set cannot be produced if an input data set contains spanned records.

If both the SYSUT1 and the SYSUT2 DD statements specify standard user labels (SUL), IEBGENER copies user labels from SYSUT1 to SYSUT2. See Appendix C, "Processing User Labels" on page 446 for a discussion of the available options for user label processing.

SYSUT2 DD Statement

The output data set for IEBGENER, as specified in SYSUT2, can contain fixed, variable, undefined, or variable spanned records (except partitioned output data sets, which cannot contain variable spanned records). These records can be reblocked by the specification of a new maximum block length on the SYSUT2 DD

statement. During reblocking, if the output data set resides on a direct access volume:

- For fixed-length or variable-length records, keys can be retained only by using the appropriate user exit.
- For variable spanned records, keys can never be retained.

If the output data set is on a card punch or a printer, the user must specify DCB information on the SYSUT2 DD statement. DCB parameters in a SYSUT2 DD statement defining an expanded partitioned data set must be compatible with the specifications made when the data set was originally created.

When RECFM, BLKSIZE, and LRECL are not specified in the JCL for the output data set, values for each are copied from the input data set's DSCB.

The output block size must always be specified when the logical record length and record format (except for U) are specified.

The default RECFM is U for the output data set. RECFM must be specified when a data set is new, a dummy data set, or a data set from a card punch or printer.

The output LRECL must be specified when editing is to be performed and the record format is FB, VS, or VBS. LRECL must also be specified when the data set is new, a dummy data set, or a data set from a card punch or printer. In all other cases, a default LRECL value is generated by IEBGENER.

If the logical record length of the output data set differs from that of the input data set, all positions in the output records must undergo editing to justify the new logical record length.

A partitioned data set cannot be produced if an input or output data set contains spanned records.

IEBGENER can not produce an output data set having a logical record length that differs from the input logical record length if both input and output RECFM are V or VB.

IEBGENER will terminate with an unpredictable message or abend code if DISP=OLD is specified on a SYSUT2 DD Statement making a specific volume request for a nonexistent data set.

SYSIN DD Statement

The SYSIN DD statement is required for each use of IEBGENER. The block size for the SYSIN data set must be a multiple of 80. Any blocking factor can be specified for this block size.

UTILITY CONTROL STATEMENTS

IEBGENER is controlled by utility control statements. The statements and the order in which they must appear are listed in Figure 59 on page 148.

The control statements are included in the control data set as required. If no utility control statements are included in the control data set, the entire input data set is copied sequentially.

When the output is to be sequential and editing is to be performed, one GENERATE statement and as many RECORD statements as required are used. If user exits are provided, an EXITS statement is used.

When the output is to be partitioned, one GENERATE statement, one MEMBER statement per output member, and RECORD statements, as required, are used. If user exits are provided, an EXITS statement is used.

Statement	Use
GENERATE	Indicates the number of member names and alias names, record identifiers, literals, and editing information contained in the control data set.
EXITS	Indicates that user routines are provided.
LABELS	Specifies user-label processing.
MEMBER	Specifies the member name and alias of a member of a partitioned data set to be created.
RECORD	Defines a record group to be processed and supplies editing information.

Figure 59. IEBGENER Utility Control Statements

Continuation requirements for utility control statements are described in "Continuing Utility Control Statements" on page 5. A nonblank character continuation mark in column 72 is optional for IEBGENER.

GENERATE Statement

The GENERATE statement is required when: (1) output is to be partitioned, (2) editing is to be performed, or (3) user routines are provided and/or label processing is specified. The GENERATE statement must appear before any other IEBGENER utility statements. If it contains errors or is inconsistent with other statements, IEBGENER is terminated.

The format of the GENERATE statement is:

[label]	GENERATE	[MAXNAME=n] [,MAXFLDS=n] [,MAXGPS=n] [,MAXLITS=n]

EXITS Statement

The EXITS statement is used to identify exit routines supplied by the user. Linkages to and from exit routines are discussed in Appendix A, "Exit Routine Linkage" on page 438.

For a detailed discussion of the processing of user labels as data set descriptors, and for discussion of user label totaling, refer to Appendix C, "Processing User Labels" on page 446.

The format of the EXITS statement is:

[label]	EXITS	[INHDR=routinename] [,OUTHDR=routinename] [,INTLR=routinename] [,OUTTLR=routinename] [,KEY=routinename] [,DATA=routinename] [,IOERROR=routinename] [,TOTAL=(routinename,size)]

LABELS Statement

The LABELS statement specifies whether or not user labels are to be treated as data by IEBGENER. For a detailed discussion of this option, refer to Appendix C, "Processing User Labels" on page 446.

The LABELS statement is used when the user wants to specify that: (1) no user labels are to be copied to the output data set, (2) user labels are to be copied to the output data set from records in the data portion of the SYSIN data set, or (3) user labels are to be copied to the output data set after they are modified by the user's label processing routines. If more than one valid LABELS statement is included, all but the last LABELS statement are ignored.

The format of the LABELS statement is:

[label]	LABELS	[DATA=YES|NO|ALL|ONLY|INPUT]

LABELS DATA=NO must be specified to make standard user labels (SUL) exits inactive when input/output data sets with nonstandard labels (NSL) are to be processed.

MEMBER Statement

The MEMBER statement is used when the output data set is to be partitioned. One MEMBER statement must be included for each member to be created by IEBGENER. The MEMBER statement provides the name and alias names of a new member.

All RECORD statements following a MEMBER statement pertain to the member named in that MEMBER statement. If no MEMBER statements are included, the output data set is organized sequentially.

The format of the MEMBER statement is:

[label]	MEMBER	NAME=(name[,alias]...)

RECORD Statement

The RECORD statement is used to define a record group and to supply editing information. A record group consists of records that are to be processed identically.

The RECORD statement is used when: (1) the output is to be partitioned, (2) editing is to be performed, or (3) user labels for the output data set are to be created from records in the data portion of the SYSIN data set. The RECORD statement defines a record group by identifying the last record of the group with a literal name.

If no RECORD statement is used, the entire input data set or member is processed without editing. More than one RECORD statement may appear in the control statement stream for IEBGENER.

Within a RECORD statement, one IDENT parameter can be used to define the record group; one or more FIELD parameters can be used to supply the editing information applicable to the record group; and one LABELS parameter can be used to indicate that this statement is followed immediately by output label records.

The format of the RECORD statement is:

[label]	**RECORD**	**[IDENT=(**length**,'**name**',**input-location**)]** **[,FIELD=([**length**]** **[,**input-location**|'**literal**']** **[,**conversion**]** **[,**output-location**])]** **[,LABELS=**n**]**

Note that the variables on the FIELD parameter are positional; that is, if any of the options are not coded, the associated comma preceding that variable must be coded.

Parameters	Applicable Control Statements	Description of Parameters
DATA	EXITS	**DATA=**routinename specifies the name of the routine that modifies the physical record (logical record for VS or VBS type records) before it is processed by IEBGENER.
	LABELS	**DATA=YES|NO|ALL|ONLY|INPUT** specifies whether user labels are to be treated as data by IEBGENER. These values can be coded: **YES** specifies that any user labels that are not rejected by a user's label processing routine are to be treated as data. Processing of labels as data ends in compliance with standard return codes. YES is the default. **NO** specifies that user labels are not to be treated as data. NO must be specified when processing input/output data sets with nonstandard labels (NSL) in order to make standard user label (SUL) exits inactive. **ALL** specifies that all user labels in the group currently being processed are to be treated as data. A return code of 16 causes IEBGENER to complete processing the remainder of the group of user labels and to terminate the job step. **ONLY** specifies that only user header labels are to be treated as data. User header labels are processed as data regardless of any return code. The job terminates upon return from the OPEN routine. **INPUT** specifies that user labels for the output data set are supplied as 80-byte input records in the data portion of SYSIN. The number of input records that should be treated as user labels must be identified by a RECORD statement.

Parameters	Applicable Control Statements	Description of Parameters
FIELD	RECORD	**FIELD=([length],[input-location|'literal'],[conversion],[output-location])** specifies field-processing and editing information. Only the contents of specified fields in the input record are copied to the output record; that is, any field in the output record that is not specified will contain meaningless information. Note that the variables on the FIELD parameter are positional; if any of the options are not coded, the associated comma preceding that variable must be coded. The values that can be coded are: length specifies the length (in bytes) of the input field or literal to be processed. If length is not specified, a length of 80 bytes is assumed. If a literal is to be processed, a length of 40 bytes or less must be specified. The length cannot exceed 8 decimal characters. input-location specifies the starting byte of the field to be processed. input-location should be coded as a decimal number. **Default:** Byte 1 is assumed. 'literal' specifies a literal (maximum length of 40 bytes) to be placed in the specified output location. If a literal contains apostrophes, each apostrophe must be written as two consecutive apostrophes. conversion specifies a 2-byte code that indicates the type of conversion to be performed on this field. If no conversion is specified, the field is moved to the output area without change. The values that can be coded are: **PZ** specifies that data (packed decimal) is to be converted to unpacked decimal data. Unpacking of the low-order digit and sign may result in an alphabetic character. **ZP** specifies that data (unpacked decimal) is to be converted to packed decimal data. **HE** specifies that data (H-set BCD) is to be converted to EBCDIC.

Parameters	Applicable Control Statements	Description of Parameters
FIELD (continued)	RECORD (continued)	**conversion (continued)** If conversion is specified in FIELD, the following restrictions apply: • PZ-type (packed-to-unpacked) conversion is impossible for packed decimal records longer than 16K bytes. • For ZP-type (unpacked-to-packed) conversion, the normal 32K-byte maximum applies. • When the ZP parameter is specified, the conversion is performed in place. The original unpacked field is replaced by the new packed field. Therefore, the ZP parameter must be omitted from subsequent references to that field. If the field is needed in its original unpacked form, it must be referenced prior to the use of the ZP parameter. If conversion is specified in the FIELD parameter, the length of the output record can be calculated for each conversion specification. When L is equal to the length of the input record, the calculation is made, as follows: • For a PZ (packed-to-unpacked) specification, 2L-1. • For a ZP (unpacked-to-packed) specification, (L/2) + C. If L is an odd number, C is 1/2; if L is an even number, C is 1. • For an (H-set BCD to EBCDIC) specification, L. output-location specifies the starting location of this field in the output records. output-location should be coded as a decimal number. The default location is byte 1.

Parameters	Applicable Control Statements	Description of Parameters
FIELD (continued)	RECORD (continued)	If both output header labels and output trailer labels are to be contained in the SYSIN data set, the user must include one RECORD statement (including the LABELS parameter), indicating the number of input records to be treated as user header labels and another RECORD statement (also including the LABELS parameter) for user trailer labels. The first such RECORD statement indicates the number of user header labels; the second indicates the number of user trailer labels. If only output trailer labels are included in the SYSIN data set, a RECORD statement must be included to indicate that there are no output header labels in the SYSIN data set (LABELS=0). This statement must precede the RECORD LABELS=n statement which signals the start of trailer label input records. For a detailed discussion of the LABELS option, refer to Appendix C, "Processing User Labels" on page 446.
IDENT	RECORD	**IDENT=(**length,'name',input-location**)** identifies the last record of the input group to which the FIELD parameters of MEMBER statement applies. If the RECORD statement is not followed by additional RECORD or MEMBER statements, IDENT also defines the last record to be processed. These values can be coded: length specifies the length (in bytes) of the identifying name. The length cannot exceed eight decimal characters. 'name' specifies the exact literal that identifies the last input record of a record group. 'name' must be coded in single apostrophes. **Default:** If no match for 'name' is found, the remainder of the input data is considered to be in one record group; subsequent RECORD and MEMBER statements are ignored. input-location specifies the starting byte of the field that contains the identifying name in the input records. input-location should be coded as a decimal number. **Default:** If IDENT is omitted, the remainder of the input data is considered to be in one record group; subsequent RECORD and MEMBER statements are ignored.
INHDR	EXITS	**INHDR=**routinename specifies the name of the routine that processes user input header labels.

Parameters	Applicable Control Statements	Description of Parameters
INTLR	EXITS	**INTLR=**routinename specifies the name of the routine that processes user input trailer labels.
IOERROR	EXITS	**IOERROR=**routinename specifies the name of the routine that handles permanent input/output error conditions.
KEY	EXITS	**KEY=**routinename specifies the name of the routine that creates the output record key. (This routine does not receive control when a data set consisting of variable spanned (VS) or variable blocked spanned (VBS) type records is processed because no processing of keys is permitted for this type of data.)
LABELS	RECORD	**LABELS=**n is an optional parameter that indicates the number of records in the SYSIN data set to be treated as user labels. The number n, which is a number from 0 to 8, must specify the exact number of label records that follow the RECORD statement. If this parameter is included, DATA=INPUT must be coded on a LABELS statement before it in the input stream.
MAXFLDS	GENERATE	**MAXFLDS=**n specifies a number that is no less than the total number of FIELD parameters appearing in subsequent RECORD statements. MAXFLDS is required if there are any FIELD parameters in subsequent RECORD statements.
MAXGPS	GENERATE	**MAXGPS=**n specifies a number that is no less than the total number of IDENT parameters appearing in subsequent RECORD statements. MAXGPS is required if there are any IDENT parameters in subsequent RECORD statements.
MAXLITS	GENERATE	**MAXLITS=**n specifies a number that is no less than the total number of characters contained in the FIELD literals of subsequent RECORD statements. MAXLITS is required if the FIELD parameters of subsequent RECORD statements contain literals. MAXLITS does not apply to literals used in IDENT parameters.

Parameters	Applicable Control Statements	Description of Parameters
MAXNAME	GENERATE	**MAXNAME=**n specifies a number that is no less than the total number of member names and aliases appearing in subsequent MEMBER statements. MAXNAME is required if there are one or more MEMBER statements.
NAME	MEMBER	**NAME=(**name[,alias]...**)** specifies a member name followed by a list of its aliases. Names of multiple members and their aliases should be coded as follows: ((name1,alias1),(name2,alias2),...) If only one name appears in the statement, it need not be enclosed in parentheses.
OUTHDR	EXITS	**OUTHDR=**routinename specifies the name of the routine that creates user output header labels. OUTHDR is ignored if the output data set is partitioned.
OUTTLR	EXITS	**OUTTLR=**routinename specifies the name of the routine that processes user output trailer labels. OUTTLR is ignored if the output data set is partitioned.
TOTAL	EXITS	**TOTAL=(**routinename,size**)** specifies that a user exit routine is to be provided prior to writing each record. The keyword OPTCD=T must be specified for the SYSUT2 DD statement. TOTAL is valid only when IEBGENER is used to process sequential data sets. These values must be coded: routinename specifies the name of the user-supplied totaling routine. size specifies the number of bytes needed to contain totals, counters, pointers, etc. size should be coded as a decimal number.

IEBGENER EXAMPLES

The examples that follow illustrate some of the uses of IEBGENER. Figure 60 can be used as a quick-reference guide to IEBGENER examples. The numbers in the "Example" column refer to the examples that follow.

Operation	Data Set Organization	Device	Comments	Example
COPY	Sequential	Card Reader and Tape	Blocked output.	1
COPY-with editing	Sequential	Card Reader and Tape	Blocked output.	2
COPY-with editing	Sequential	Card Reader and Tape	Blocked output. Input includes //cards.	3
COPY-with editing	Sequential	Card Reader and Disk	Blocked output. Input includes // cards.	4
PRINT	Sequential	Card Reader and Printer	Input includes // cards. System output device is a printer.	5
CONVERT	Sequential input, Partitioned output	Tape and Disk	Blocked output. Three members are to be created.	6
COPY-with editing	Sequential	Disk	Blocked output. Two members are to be merged into existing data set.	7
COPY-with editing	Sequential	Tape	Blocked output. Data set edited as one record group.	8
COPY-with editing	Sequential	Disk	Blocked output. New record length specified for output data set. Two record groups specified.	9
COPY-with editing	Sequential	Tape	Blocked output. Data set edited as one record group.	10

Figure 60. IEBGENER Example Directory

Examples that use **disk** or **tape** in place of actual device numbers must be changed before use. See "DASD and Tape Device Support" on page 3 for valid device number notation.

IEBGENER EXAMPLE 1

In this example, a card-input, sequential data set is copied to a 9-track tape volume.

The example follows:

```
//CDTOTAPE JOB  09#660,SMITH
//         EXEC PGM=IEBGENER
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   DUMMY
//SYSUT2   DD   DSNAME=OUTSET,UNIT=tape,LABEL=(,SL),
//               DISP=(,KEEP),VOLUME=SER=001234,
//               DCB=(RECFM=FB,LRECL=80,BLKSIZE=2000)
//SYSUT1   DD   *

(input card data set)

/*
```

The job control statements are discussed below:

- SYSIN DD defines a dummy data set. No editing is performed; therefore, no utility control statements are needed.
- SYSUT2 DD defines the output data set, OUTSET. The data set is written to a tape volume with IBM standard labels. The data set is to reside as the first (or only) data set on the volume.
- SYSUT1 DD defines the card-input data set. The data set contains no // or /* cards.

IEBGENER EXAMPLE 2

In this example, a card-input, sequential data set is to be copied to a tape volume. The control data set is a member of a partitioned data set.

```
//CDTOTAPE JOB  09#660,SMITH
//         EXEC PGM=IEBGENER
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   DSNAME=CNTRLIBY(STMNTS),UNIT=disk,
//               DISP=(OLD,KEEP),VOLUME=SER=111112,
//               DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//SYSUT2   DD   DSNAME=OUTSET,UNIT=tape,LABEL=(,SL),
//               DCB=(RECFM=FB,LRECL=80,BLKSIZE=2000),
//               DISP=(,KEEP),VOLUME=SER=001234
//SYSUT1   DD   *

(input card data set)

/*
```

The job control statements are discussed below:

- SYSIN DD defines the control data set, which contains the utility control statements. The control statements reside as a member, STMNTS, in a partitioned data set called CNTRLIBY.
- SYSUT2 DD defines the output data set, OUTSET. The data set is written as the first data set on the tape volume.

- SYSUT1 DD defines the card-input data set. The data set can contain no // cards, since SYSUT1 has not been specified as DATA.

IEBGENER EXAMPLE 3

In this example, a card-input, sequential data set is copied to a tape volume. The input contains cards that have slashes (//) in columns 1 and 2. The control data set is a member of a partitioned data set.

```
//CDTOTAPE JOB  09#660,SMITH
//         EXEC PGM=IEBGENER
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   DSNAME=CNTRLIBY(STMNTS),UNIT=disk,
//              DISP=(OLD,KEEP),VOLUME=SER=111112
//SYSUT2   DD   DSNAME=OUTSET,UNIT=tape,LABEL=(2,SL),
//              VOLUME=SER=001234,DCB=(RECFM=FB,LRECL=80,
//              BLKSIZE=2000),DISP=(,KEEP)
//SYSUT1   DD   DATA

(input card data set, including // cards)

/*
```

The job control statements are discussed below:

- SYSIN DD defines the data set containing the utility control statements. The statements reside as a member, STMNTS, in a partitioned data set called CNTRLIBY.
- SYSUT2 DD defines the copied sequential data set (output), called OUTSET. The data set is written as the second data set on the specified tape volume.
- SYSUT1 DD defines the card-input data set. The data set is to be edited as specified in the utility control statements (not shown). The input data set contains // cards.

IEBGENER EXAMPLE 4

In this example, a card-input, sequential data set is copied to a disk volume. The input data set contains // cards.

```
//CDTODISK JOB  09#660,SMITH
//         EXEC PGM=IEBGENER
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   DSNAME=CNTRLIBY(STMNTS),UNIT=disk,
//              DISP=(OLD,KEEP),VOLUME=SER=111112
//SYSUT2   DD   DSNAME=OUTSET,UNIT=disk,VOLUME=SER=111113,
//              DISP=(,KEEP),SPACE=(TRK,(10,10)),
//              DCB=(RECFM=FB,LRECL=80,BLKSIZE=2000)
//SYSUT1   DD   DATA

(input card data set, including // cards)

/*
```

The job control statements are discussed below:

- SYSIN DD defines the control data set, which contains the utility control statements. The control statements reside as a member, STMNTS, in a partitioned data set.
- SYSUT2 DD defines the output data set. Ten tracks of primary storage space and ten tracks of secondary space are allocated for the data set on a disk volume.
- SYSUT1 DD defines the card-input data set. The data set is to be edited as specified in the utility control statements (not shown).

IEBGENER EXAMPLE 5

In this example, the content of a card data set is printed. The printed output is left-aligned, with one 80-byte record appearing on each line of printed output.

```
//CDTOPTR  JOB  09#660,SMITH
//         EXEC PGM=IEBGENER
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   DUMMY
//SYSUT2   DD   SYSOUT=A,DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//SYSUT1   DD   DATA
(input card data set, including // cards)
/*
```

The job control statements are discussed below:

- SYSIN DD defines a dummy data set. No editing is performed; therefore, no utility control statements are required.
- SYSUT2 DD indicates that the output is to be written on the system output device (printer). Carriage control can be specified by changing the RECFM=F subparameter to RECFM=FA.
- SYSUT1 DD defines the input card data set. The input data set contains // cards.

IEBGENER EXAMPLE 6

In this example, a partitioned data set (consisting of three members) is created from sequential input.

```
//TAPEDISK JOB  09#660,SMITH
//         EXEC PGM=IEBGENER
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=INSET,UNIT=tape,LABEL=(,SL),
//              DISP=(OLD,KEEP),VOLUME=SER=001234
//SYSUT2   DD   DSNAME=NEWSET,UNIT=disk,DISP=(,KEEP),
//              VOLUME=SER=111112,SPACE=(TRK,(10,5,5)),
//              DCB=(RECFM=FB,LRECL=80,BLKSIZE=2000)
//SYSIN    DD   *
     GENERATE  MAXNAME=3,MAXGPS=2
       MEMBER  NAME=MEMBER1
GROUP1 RECORD  IDENT=(8,'FIRSTMEM',1)
       MEMBER  NAME=MEMBER2
GROUP2 RECORD  IDENT=(8,'SECNDMEM',1)
       MEMBER  NAME=MEMBER3
/*
```

The control statements are discussed below:

- SYSUT1 DD defines the input data set (INSET). The data set is the first data set on a tape volume.
- SYSUT2 DD defines the output partitioned data set (NEWSET). The data set is to be placed on a disk volume. Ten tracks of primary space, five tracks of secondary space, and five blocks (256 bytes each) of directory space are allocated to allow for future expansion of the data set. The output records are blocked to reduce the space required by the data set.
- SYSIN DD defines the control data set, which follows in the input stream. The utility control statements are used to create members from sequential input data; the statements do not specify any editing.
- GENERATE indicates that: (1) three member names are included in subsequent MEMBER statements and (2) the IDENT parameter appears twice in subsequent RECORD statements.
- The first MEMBER statement assigns a member name (MEMBER1) to the first member.
- The first RECORD statement (GROUP1) identifies the last record to be placed in the first member. The name of this record (FIRSTMEM) appears in bytes 1 through 8 of the input record.
- The remaining MEMBER and RECORD statements define the second and third members. Note that, as there is no RECORD statement associated with the third MEMBER statement, the remainder of the input file will be loaded as the third member.

IEBGENER EXAMPLE 7

In this example, sequential input is converted into two partitioned members. The newly created members are merged into an existing partitioned data set. User labels on the input data set are passed to the user exit routine.

```
//DISKTODK JOB  09#660,SMITH
//         EXEC PGM=IEBGENER
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=INSET,UNIT=disk,DISP=(OLD,KEEP),
//              VOLUME=SER=111112,
//              LABEL=(,SUL)
//SYSUT2   DD   DSNAME=EXISTSET,UNIT=disk,DISP=(MOD,KEEP),
//              VOLUME=SER=111113
//SYSIN    DD   *
     GENERATE   MAXNAME=3,MAXGPS=1
        EXITS   INHDR=ROUT1,INTLR=ROUT2
       MEMBER   NAME=(MEMX,ALIASX)
GROUP1 RECORD   IDENT=(8,'FIRSTMEM',1)
       MEMBER   NAME=MEMY
/*
```

The control statements are discussed below:

- SYSUT1 DD defines the input data set (INSET). The input data set, which resides on a disk volume, has standard and user labels.
- SYSUT2 DD defines the output partitioned data set (EXISTSET). The members created during this job step are merged into the partitioned data set.

- SYSIN DD defines the control data set, which follows in the input stream. The utility control statements are used to create members from sequential input data; the statements do not specify any editing.
- GENERATE indicates that: (1) a maximum of three names and aliases are included in subsequent MEMBER statements and (2) one IDENT parameter appears in a subsequent RECORD statement.
- EXITS defines the user routines that are to process user labels.
- The first MEMBER statement assigns a member name (MEMX) and an alias (ALIASX) to the first member.
- The first RECORD statement (GROUP1) identifies the last record to be placed in the first member. The name of this record (FIRSTMEM) appears in bytes 1 through 8 of the input record.
- The second MEMBER statement assigns a member name (MEMY) to the second member. The remainder of the input data set is included in this member.

IEBGENER EXAMPLE 8

In this example, a sequential input data set is edited and copied.

```
                                                                      72
//TAPETAPE JOB  09#660,SMITH
//         EXEC PGM=IEBGENER
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=OLDSET,UNIT=tape,DISP=(OLD,KEEP),
//              VOLUME=SER=001234,LABEL=(3,SL)
//SYSUT2   DD   DSNAME=NEWSET,UNIT=tape,DISP=(NEW,PASS),
//              DCB=(RECFM=FB,LRECL=80,BLKSIZE=2000),
//              VOLUME=SER=001235,LABEL=(,SL)
//SYSIN    DD   *
     GENERATE   MAXFLDS=3,MAXLITS=11
       RECORD   FIELD=(10,'**********',,1),                          C
                FIELD=(5,1,HE,11),FIELD=(1,'=',,16)
        EXITS   INHDR=ROUT1,OUTTLR=ROUT2
       LABELS   DATA=INPUT
       RECORD   LABELS=2

(first header label record)
(second header label record)

       RECORD   LABELS=2

(first trailer label record)
(second trailer label record)

/*
```

The control statements are discussed below:

- SYSUT1 DD defines the sequential input data set (OLDSET). The data set was originally written as the third data set on a tape volume.
- SYSUT2 DD defines the sequential output data set (NEWSET). The data set is written as the first data set on a tape volume. The output records are blocked to reduce the space required by the data set and to reduce the access time

required when the data set is subsequently referred to. The data set is passed to a subsequent job step.

- SYSIN DD defines the control data set, which follows in the input stream.
- GENERATE indicates that: (1) a maximum of three FIELD parameters is included in subsequent RECORD statements and (2) a maximum of 11 literal characters are included in subsequent FIELD parameters.
- The first RECORD statement controls the editing, as follows: (1) asterisks are placed in positions 1 through 10, (2) bytes 1 through 5 of the input record are converted from H-set BCD to EBCDIC mode and moved to positions 11 through 15, and (3) an equal sign is placed in byte 16.
- EXITS indicates that the specified user routines require control when SYSUT1 is opened and when SYSUT2 is closed.
- LABELS indicates that labels are included in the input stream.
- The second RECORD statement indicates that the next two records from SYSIN should be written out as user header labels on SYSUT2.
- The third RECORD statement indicates that the next two records from SYSIN should be written as user trailer labels on SYSUT2.

This example shows the relationship between the RECORD LABELS statement, the LABELS statement, and the EXITS statement. IEBGENER attempts to write a first and second label trailer as user labels at close time of SYSUT2 before returning control to the system; the user routine, ROUT2, can review these records and change them, if necessary.

IEBGENER EXAMPLE 9

In this example, a sequential input data set is edited and copied.

```
                                                                      72
//DISKDISK JOB  09#660,SMITH
//         EXEC PGM=IEBGENER
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=OLDSET,UNIT=disk,DISP=(OLD,KEEP),
//              VOLUME=SER=111112
//SYSUT2   DD   DSNAME=NEWSET,UNIT=disk,DISP=(NEW,KEEP),
//              VOLUME=SER=111113,DCB=(RECFM=FB,LRECL=80,
//              BLKSIZE=640),SPACE=(TRK,(20,10))
//SYSIN    DD   *
     GENERATE   MAXFLDS=4,MAXGPS=1
        EXITS   IOERROR=ERRORRT
GROUP1 RECORD   IDENT=(8,'FIRSTGRP',1),                             C
                FIELD=(21,80,,60),FIELD=(59,1,,1)
GROUP2 RECORD   FIELD=(11,90,,70),FIELD=(69,1,,1)
/*
```

The control statements are discussed below:

- SYSUT1 DD defines the input data set (OLDSET).
- SYSUT2 DD defines the output data set (NEWSET). Twenty tracks of primary storage space and ten tracks of secondary storage space are allocated for the data set on a disk

volume. The logical record length of the output records is 80 bytes, and the output is blocked.

- SYSIN DD defines the control data set, which follows in the input stream.
- GENERATE indicates that: (1) a maximum of four FIELD parameters are included in subsequent RECORD statements and (2) one IDENT parameter appears in a subsequent RECORD statement.
- EXITS identifies the user routine that handles input/output errors.
- The first RECORD statement (GROUP1) controls the editing of the first record group, as follows: (1) FIRSTGRP, which appears in bytes 1 through 8 of an input record, is defined as being the last record in the first group of records and (2) bytes 80 through 100 of each input record are moved into positions 60 through 80 of each corresponding output record. (This example implies that bytes 60 through 79 of the input records in the first record group are no longer required; thus, the logical record length is shortened by 20 bytes.) The remaining bytes within each input record are transferred directly to the output records, specified in the second FIELD parameter.
- The second RECORD statement (GROUP2) indicates that the remainder of the input records are to be processed as the second record group. Bytes 90 through 100 of each input record are moved into positions 70 through 80 of the output records. (This example implies that bytes 70 through 89 of the input records from group 2 are no longer required; thus, the logical record length is shortened by 20 bytes.) The remaining bytes within each input record are transferred directly to the output records, specified in the second FIELD parameter.

If the logical record length of the output data set differs from that of the input data set (as in this example), all positions in the output records must undergo editing to justify the new logical record length.

IEBGENER EXAMPLE 10

In the example, a sequential input data set is edited and copied.

```
                                                                      72
//TAPETAPE JOB  ...
//         EXEC PGM=IEBGENER
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=OLDSET,UNIT=tape,DISP=(OLD,KEEP),
//              VOLUME=SER=001234,LABEL=(3,SUL)
//SYSUT2   DD   DSNAME=NEWSET,UNIT=tape,DISP=(NEW,PASS),
//              VOLUME=SER=001235,LABEL=(,SUL),
//              DCB=(RECFM=FB,LRECL=80,BLKSIZE=2000)
//SYSIN    DD   *
     GENERATE   MAXFLDS=3,MAXLITS=11
       RECORD   FIELD=(10,'**********',,1),                          C
                FIELD=(5,1,HE,11),FIELD=(1,'=',,16)
       LABELS   DATA=INPUT
       RECORD   LABELS=3

(first header label record)
(second header label record)
(third header label record)

       RECORD   LABELS=2

(first trailer label record)
(second trailer label record)

/*
```

The control statements are discussed below:

- SYSUT1 DD defines the input data set (OLDSET). The data set is the third data set on a tape volume.
- SYSUT2 DD defines the output data set (NEWSET). The data set is written as the first or only data set on a tape volume. The output records are blocked to reduce the space required by the data set and to reduce the access time required when the data set is subsequently referred to. The data set is passed to a subsequent job step.
- SYSIN DD defines the control data set, which follows in the input stream.
- GENERATE indicates that: (1) a maximum of three FIELD parameters are included in subsequent RECORD statements and (2) a maximum of 11 literal characters are included in subsequent FIELD parameters.
- The first RECORD statement controls the editing, as follows: (1) asterisks are placed in positions 1 through 10, (2) bytes 1 through 5 of the input record are converted from H-set BCD to EBCDIC mode and moved to positions 11 through 15, and (3) an equal sign is placed in byte 16.
- LABELS indicates that label records are included in the input stream.
- The second RECORD statement indicates that three 80-byte records (cards), to be written as user labels on the output data set, immediately follow.
- The third RECORD statement indicates that the following cards are to be treated as trailer labels.

IEBIMAGE PROGRAM

IEBIMAGE is a data set utility that creates and maintains the following types of IBM 3800 Printing Subsystem and IBM 4248 Printer modules and stores them in a library:

- Forms control buffer modules for the 3800 and 4248 that specify controls for the vertical line spacing and any one of 12 channel codes per line.
- Copy modification modules for the 3800 that specify data that is to be printed on every page for specified copies of the output data set.
- Character arrangement table modules for the 3800 that translate the input data into printable characters and identify the associated character set(s) and graphic character modification module(s).
- Graphic character modification modules for the 3800 that contain the scan patterns of user-designed characters and/or characters from IBM-supplied modules.
- Library character set modules for the 3800 that contain the scan patterns of IBM-supplied character sets and/or user-defined character sets.

The IEBIMAGE program creates and maintains all modules required for use on the 3800 Model 1 and Model 3 printers. The program default is to build these modules in the 3800 Model 1 format; however, 3800 Model 3 compatibility can be specified with IEBIMAGE utility control statements.

IEBIMAGE can also be used to create and maintain FCB modules for the 4248 printer.

3262 Model 5 Printer

The 4248 FCB modules created by IEBIMAGE are compatible with the 3262 Model 5 Printer; however, the 3262 Model 5 does not support variable printer speeds or the horizontal copy feature of the 4248. Unless otherwise stated, where a reference to the 4248 printer is used in this chapter, the 3262 Model 5 can be substituted.

End of 3262 Model 5 Printer

GENERAL INFORMATION

STORAGE REQUIREMENTS

For IEBIMAGE

The IEBIMAGE utility program is IBM-supplied and requires pageable virtual storage in which to operate. The storage needed by IEBIMAGE is given by the formula:

Storage requirements (in bytes) = 44K+4B+H

B The largest block size in the job step, rounded to the next highest multiple of 2K. If the format specified for the data set is VS, and LRECL is less than 32K, then B is the maximum logical record length, rounded to the next highest multiple of 2K.

H The size of the largest member to be loaded from SYS1.IMAGELIB, rounded to the next highest multiple of 2K.

K 1024 bytes.

For SYS1.IMAGELIB

The auxiliary storage requirement in tracks for SYS1.IMAGELIB is:

Number of tracks = (A+B)/T

A The number of 1403 UCS images, 3211 UCS images, 3211 FCB images, 3525 data protection images, 3886 format records, 3890 SCI programs, 3800 FCB modules, 4248 FCB images, 3262 Model 5 FCB images, and 3800 character arrangement tables (both IBM-supplied and user-defined images or modules, as applicable).

If the appropriate printer is in the system, IBM supplies twelve 1403 UCS images, five 3211 UCS images, four 3211 FCB images, one 3800 FCB image, one 4245 UCS image table, one 4248 UCS image table, and fourteen 3800 character arrangement tables. According to the TABLE parameter coded on the DATAMGT system generation macro, IBM supplies the following number of additional character arrangement tables:

- 5 if T3211 is specified
- 13 if T1403 is specified
- 10 if TOCR is specified
- 3 if TKAT is specified
- 3 if TFMT is specified

If TABLE = ALL is coded, add all the above numbers. If ALL, T3211, or T1403 is coded, add two more tables for the GRAFSPC1 and GRF2SPC1 graphic character modification modules.

Note that IBM supplies no 4245 or 4248 UCS images in SYS1.IMAGELIB. The 4245 and 4248 printers load their own UCS images into the UCS buffer at power-on time. IBM does supply 4245 and 4248 FCB images which may be used. For more information on printer-supplied UCS or FCB images, see System Programming Library: Data Management.

B (V+600)/1500 for each 3800 graphic character modification module and library character set module, each 3800 copy modification module, 4245 UCS image table, 4248 UCS image table, and each 3890 SCI program that is more than approximately 600 bytes. V is the virtual storage requirement in bytes for each module. The virtual storage requirements for the IBM-supplied 3800 graphic character modification module containing the World Trade National Use Graphics are 32420 bytes for Model 1 and 55952 bytes for Model 3. The virtual storage requirements for the IBM-supplied 3800 library character sets for the Model 1 are 4680 bytes and 8064 bytes for the Model 3.

T The approximate number of members per track, depending on type of volume. Because of the overhead bytes and blocks in a load module, the difference in space requirements for an 80-byte module and a 400-byte module is small. These constants assume an average member of 8 blocks, including a file mark, with a total data length of 800 bytes. For example, on a 3330 with 135 bytes of

block overhead, the assumed average is 1880 bytes. If a different average member data length and average number of blocks per member are anticipated, these constants should reflect the actual number of members per track. To determine the number of members per track, divide the average member length, including block overhead, into the track capacity for the device. (Track capacity for DASD is discussed in Data Management Macro Instructions.)

```
T = 3 for a 2305-1
    6 for a 2305-2
    4 for a 2314/2319
    7 for a 3330 or a 3330-11
    4 for a 3340 or 3344
    8 for a 3350
    8 for a 3375
    9 for a 3380
```

The result, (A+B)/T, is the track requirement.

The number of directory blocks for SYS1.IMAGELIB is given by the formula:

```
Number of directory blocks = (A+C+D)/6
```

A As calculated to determine the track requirement, above.

C The number of modules used to calculate B, when calculating the track requirement.

D The number of aliases. The IBM-supplied 1403 UCS images have four aliases and the IBM-supplied 3211 UCS images have six aliases. If they will not be used, these aliases can be scratched after system generation.

MAINTAINING THE SYS1.IMAGELIB DATA SET

You will normally maintain SYS1.IMAGELIB using several programs in conjunction with IEBIMAGE. For example, you may find it necessary to rename or delete modules or to compress or list the entire contents of the data set. Utility programs such as IEBCOPY, IEBPTPCH, IEHLIST, IEHMOVE, and IEHPROGM (as described in this book) and HMASPZAP or AMASPZAP (as described in Service Aids) should be used to help maintain SYS1.IMAGELIB.

If you use programs other than IEBIMAGE for maintenance, you must specify the full module name. The module's full name consists of a 4-character prefix followed by its 1- to 4-character user-assigned name. It is thus a 5- to 8-character member name in the form:

FCB2xxxx, which identifies an FCB module that may be used with a 3203, 3211, 3262 Model 5, 4248, or 4245 printer. Note that the 4248 accepts FCBs that will also work with a 3203, 3211, 3262 Model 5, or 4245 printer.

FCB3xxxx, which identifies a 3800 FCB module

FCB4xxxx, which identifies an FCB module that may be used with a 4248 or 3262 Model 5 printer

MOD1xxxx, which identifies a 3800 copy modification module

XTB1xxxx, which identifies a 3800 character arrangement table module

GRAFxxxx, which identifies a graphic character modification module for a 3800 Model 1

GRF2xxxx, which identifies a graphic character modification module for a 3800 Model 3

LCS1nn, which identifies a library character set module for a 3800 Model 1

LCS2nn, which identifies a library character set module for a 3800 Model 3

where:

xxxx
is the 1- to 4-character user-assigned name of the module.

nn
is the 2-character user-assigned ID of the module.

Alias names are not supported by IEBIMAGE, so you should be careful if you use them. For example, if you change a module by specifying its alias name, the alias name becomes the main name of the new module, and the old module is no longer accessible via the alias but is still accessible via its original main name.

GENERAL MODULE STRUCTURE

Each module contains eight bytes of header information preceding the data. For the 3800 printing subsystem, the general module header is shown in Figure 61.

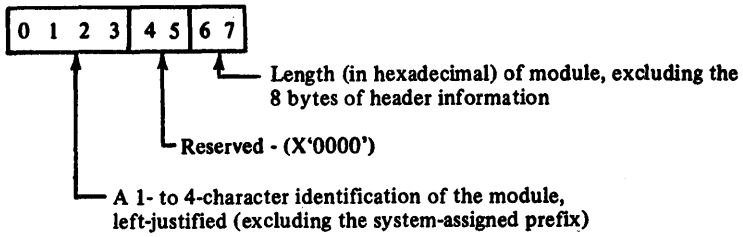


Figure 61. 3800 General Module Header

Header information for the 4248 printer FCB module is shown, with the module format, in Figure 62.1 on page 172.

The SETPRT SVC uses the name to:

- Identify the module in the image library
- Store the name in the UCB extension

The SETPRT SVC uses the length to:

- Obtain sufficient storage for the module
- Build channel programs to load the data into the printer

NAMING CONVENTIONS FOR MODULES

Each module placed in a library by the IEBIMAGE utility has a 4-character system-assigned prefix as the first part of its name. These prefixes are:

FCB3 for 3800 forms control buffer modules

FCB4 for 3262 Model 5 and 4248 forms control buffer modules

MOD1 for 3800 copy modification modules

XTB1 for 3800 character arrangement table modules

GRAF for graphic character modification modules for a 3800 Model 1

GRF2 for graphic character modification modules for a 3800 Model 3

LCS1 for library character set modules for a 3800 Model 1

LCS2 for library character set modules for a 3800 Model 3

You can assign a 1- to 4-character identifier (name) to the module you create by using the NAME control statement in the operation group you use to build the module. If the module is a library character set, the ID assigned to it must be exactly two characters. Each of those characters must be within the range 0 through 9, and A through F; the second character must represent an odd hexadecimal digit. However, the combinations X'7F' and X'FF' are not allowed. Except for library character set modules, this identifier is used in the JCL, the SETPRT parameter, or the character arrangement table to identify the module to be loaded.

While IEBIMAGE refers only to the 1- to 4-character name or the 2-character ID (the suffix) that is appended to the prefix, the full name must be used when using other utilities (such as IEBPTPCH or IEHPROGM).

USING IEBIMAGE

CREATING A FORMS CONTROL BUFFER MODULE

The forms control buffer (FCB) module is of variable length and contains vertical line spacing information (6, 8, or 12 lines per inch for the 3800 Model 1; 6 or 8 lines per inch for the 4248; and 6, 8, 10, or 12 lines per inch for the 3800 Model 3). The FCB module can also identify one of 12 carriage-control channel codes for each line. For the 4248 printer, the module also contains information on the horizontal copy feature and the printer speed.

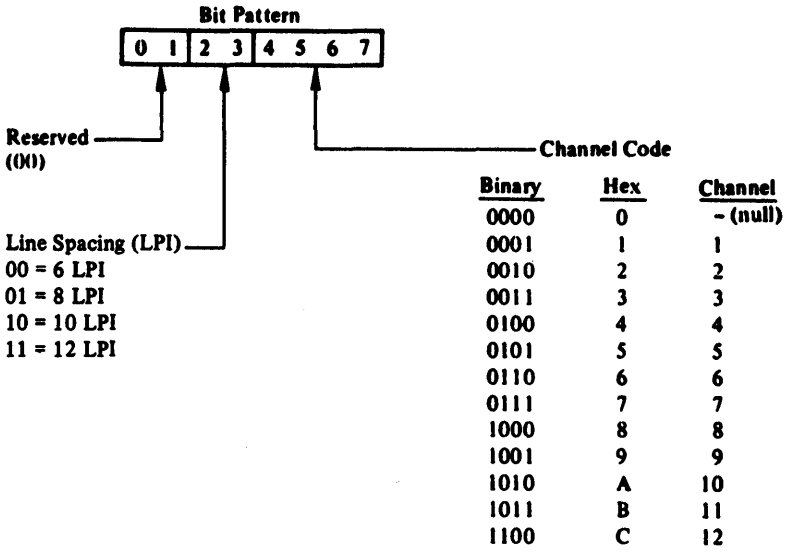
The FCB module is created and stored in an image library, using the FCB and NAME utility control statements of the IEBIMAGE program. For the 4248 FCB module, the INCLUDE and OPTION statements can also be coded to indicate that an existing FCB module (prefix FCB2 or FCB4) is to be used as a model.

For the 3800, IBM supplies one default FCB image in SYS1.IMAGELIB, called FCB3STD1. For the 4248, although the last FCB image loaded is reloaded by the printer at power-on time, IBM supplies two FCB images that may also be used by printers other than the 4248. For the 3262 Model 5, a default FCB image is also supplied.

3800 FCB Module Structure

The FCB data following the header information is a series of 1-byte line control codes for each physical line of the form. There are 18 to 144 of these bytes, depending on the length of the form.

Each byte is a bit pattern describing one of 12 channel codes for vertical forms positioning and one of four lines-per-inch codes for vertical line spacing. The structure of the 3800 FCB module is shown in Figure 62.



Binary	Hex	Channel
0000	0	- (null)
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12

Figure 62. 3800 FCB Module Structure

- The top and bottom 1/2 inch of each page are unprintable, and the bytes corresponding to these positions must be void of any channel codes. Three bytes of binary zeros are supplied by the IEBIMAGE utility for the top and bottom 1/2 inch.
- The total number of lines defined in the module must be equal to the length of the form. The printable lines defined must start 1/2 inch below the top and stop 1/2 inch from the bottom of the form.

4248 FCB Module Structure

The FCB data following the header information consists of at least five bytes: a flag byte (X'7E'), a control byte (containing information about the horizontal copy feature and printer speed), an offset byte, one or more FCB data bytes (similar to the 3800 data byte for each physical line of the form), and an end-of-sheet byte (X'FE'). The format of the 4248 FCB module is shown in Figure 62.1 on page 172.

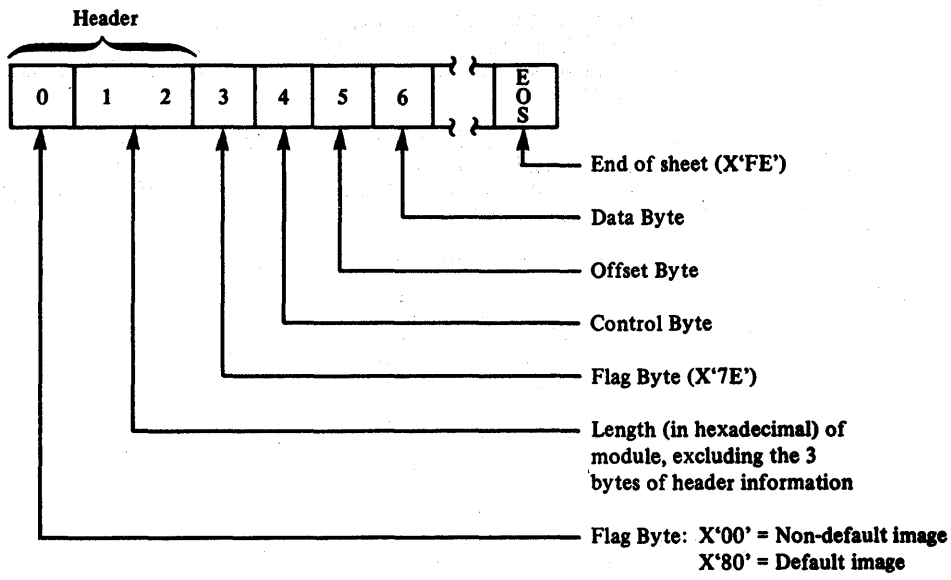


Figure 62.1. 4248 FCB Module Structure

The control byte is a bit pattern describing whether the horizontal copy feature is active and what printer speed is to be set when the FCB is loaded into the buffer. The structure of the control byte is shown in Figure 62.2.

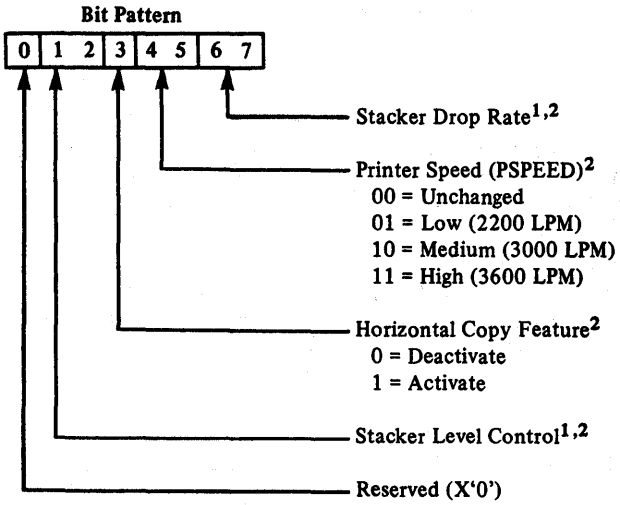


Figure 62.2. 4248 FCB Module Control Byte

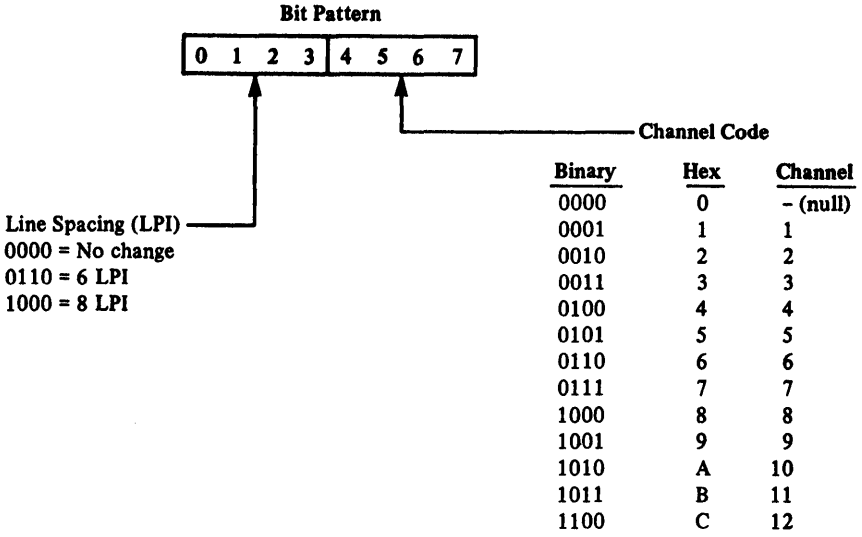
Notes to Figure 62.2:

1 IEBIMAGE sets these bits to zero. For more information on the stacker drop rate and stacker level control bits, see the appropriate hardware manual for your printer.

2 If the module is used by a 3262 Model 5 printer, these bits are ignored.

The offset byte follows the control byte and is set either to zero or to the print position of the horizontal copy (2 through 168).

The data byte is a bit pattern similar to that produced for the 3800 printing subsystem. Each data byte describes one of 12 channel codes for vertical forms positioning and one of the allowed lines-per-inch codes for vertical line spacing. The structure of the data byte is shown in Figure 62.3.



Binary	Hex	Channel
0000	0	- (null)
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12

Figure 62.3. 4248 FCB Module Data Byte

The total number of lines defined in the module must be equal to the length of the form.

FCB Module Listing

Figure 63 on page 172.2 shows the IEBIMAGE listing of a 3800 FCB module. The notes that follow the figure describe the encircled numbers in the figure.

For the 4248 FCB module, the IEBIMAGE listing also includes the horizontal copy feature, printer speed setting, and default settings.

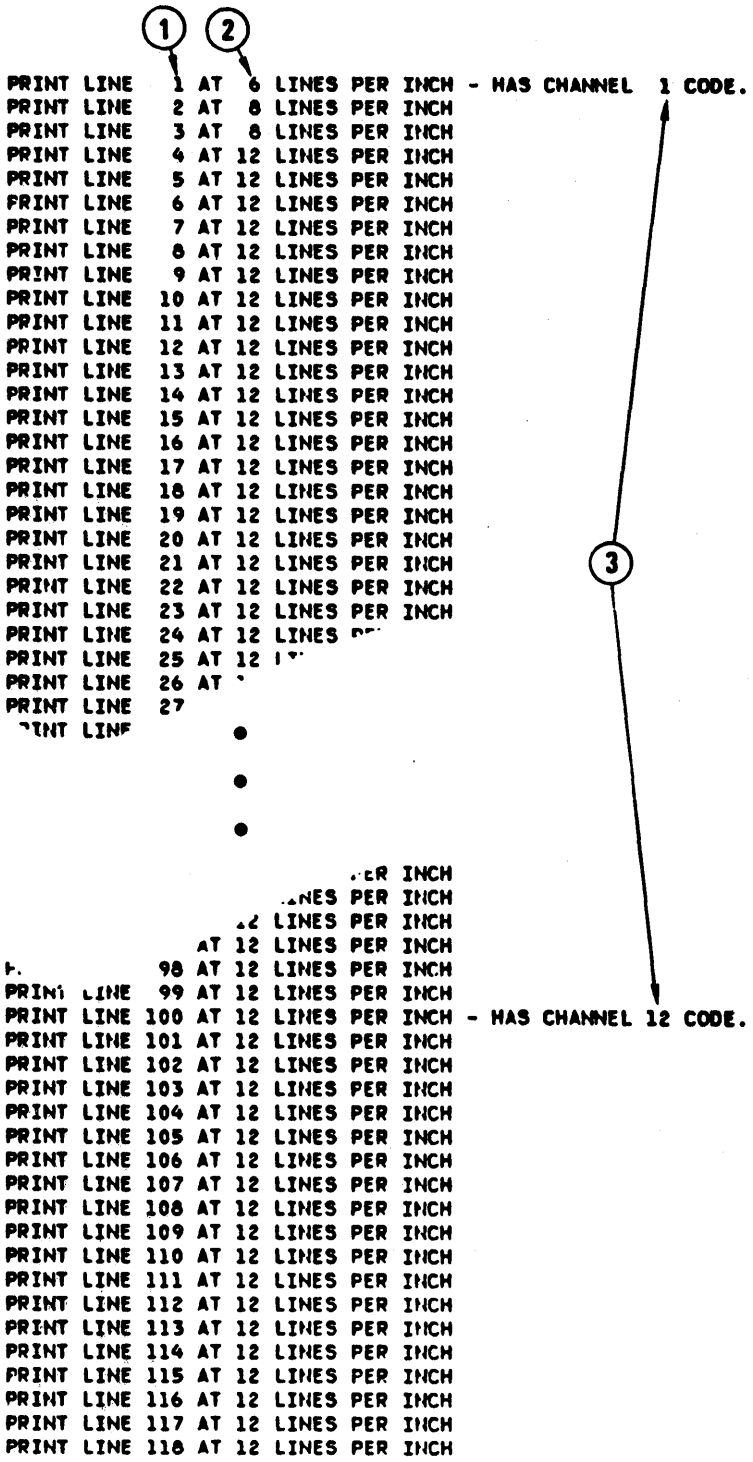


Figure 63. IEBIMAGE Listing of a Forms Control Buffer Module

Notes to Figure 63:

1. The line number. Each line of the form is listed in this fashion.
2. The vertical spacing of the line, in lines per inch.
3. The channel code, printed for each line that includes a channel code.

CREATING A COPY MODIFICATION MODULE

The 3800 copy modification module contains predefined data for modifying some or all copies of an output data set. Segments of the module contain predefined text, its position on each page of the output data set, and the copy or copies the text applies to.

The copy modification module is created and stored in an image library using the INCLUDE, OPTION, COPYMOD, and NAME utility control statements of IEBIMAGE.

The INCLUDE statement identifies a module that is to be copied and used as a basis for the newly created module. The OPTION statement with the OVERRUN parameter allows the user to suppress the printing of line overrun condition messages for those vertical line spacings that are not applicable to the job. The OPTION statement with the DEVICE parameter specifies 3800 Model 3 compatibility mode processing. The COPYMOD statement is used to describe the contents of one of the new module's segments. The NAME statement is used to identify the new module and to indicate whether it is new or is to replace an existing module with the same name.

COPYMOD Module Structure

The copy modification data following the header information is a series of segments. Each segment is of variable length and is composed of the components shown in Figure 64.

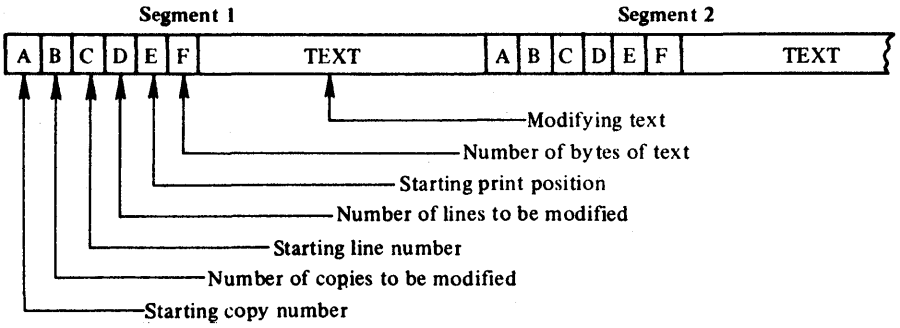


Figure 64. Copy Modification Module Structure

A, B, C, D, E, and F are each 1-byte fields.

- If the module contains more than one segment, the starting copy number must be equal to or greater than the starting copy number in the previous segment.
- Any string of the same character within the text may be compressed into 3 bytes. The first such byte is X'FF', the

second byte is the number of compressed characters, and the third byte is the data code for the character.

- The size of the module is limited to 8192 bytes of data and 8 bytes of header information.

COPYMOD Module Listing

Figure 65 shows the listing of three segments of a copy modification module. This listing shows only the positioning of the modifying text. To print out the text itself, you can use the IEBPTPCH utility program. The numbered notes that follow the figure describe the items marked with the circled numbers.

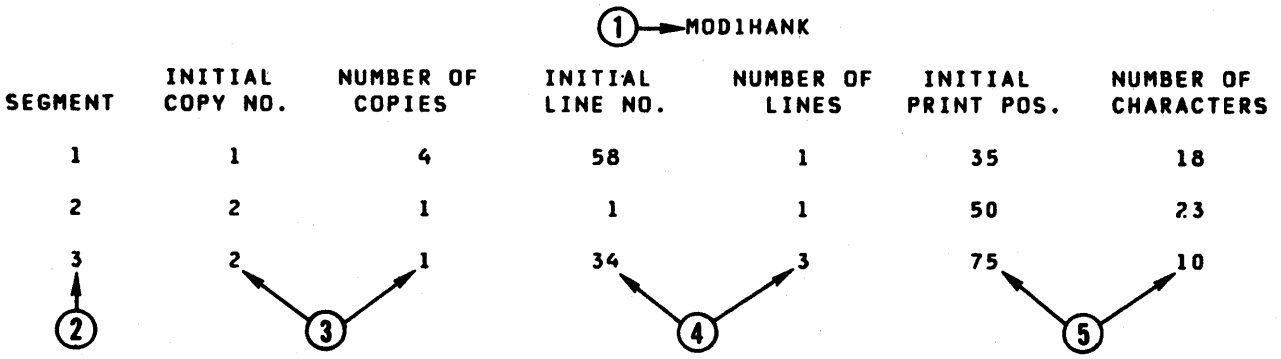


Figure 65. IEBIMAGE Listing of Three Segments of a Copy Modification Module

Notes to Figure 65:

In this example, each note refers to the module's third segment.

1. The name of the copy modification module as it exists in the SYS1.IMAGELIB data set's directory (including the 4-byte system-assigned prefix).
2. The segment number of the modification segment.
3. This segment applies only to the second copy of the output data set.
4. The text of the segment is located on lines 34, 35, and 36.
5. The text on each line starts at the 75th character, and occupies 10 character spaces.

CREATING A CHARACTER ARRANGEMENT TABLE MODULE

The 3800 character arrangement table module is fixed length and consists of three sections:

- System control information, which contains the module's name and length.
- The translate table, which contains 256 one-byte translate table entries, corresponding to the 8-bit data codes (X'00' through X'FF'). A translate table entry can identify one of 64 character positions in any one of four writable character generation modules (WCGMs) except the last position in the fourth WCGM (WCGM 3), which would be addressed by X'FF'. The code X'FF' is reserved to indicate an unprintable

character. When an entry of X'FF' is detected by the printer as a result of attempting to translate an invalid 8-bit data code, the printer prints a blank and sets the data-check indicator on (unless the block-data-check option is in effect).

- Identifiers, which identify the character sets and the graphic character modification modules associated with the character arrangement table.

The character arrangement table is created using the INCLUDE, TABLE, and NAME utility control statements. The INCLUDE statement identifies an existing character arrangement table that is to be copied and used as a basis for the new module. The TABLE statement describes the new or modified module's contents. The NAME statement identifies the character arrangement table and indicates whether it is new or is to replace an existing module with the same name.

The OPTION statement with the DEVICE=3800M3 parameter should be specified when printing an existing character arrangement table for a 3800 Model 3 to ensure that the system assigns the correct prefix to the graphic modification module name associated with the character arrangement table.

See IBM 3800 Printing Subsystem Programmer's Guide for information on IBM-supplied character arrangement tables and character sets.

Note: All characters in a character set might not be referred to by the character arrangement table you select. The character arrangement table corresponds to a print train, which is sometimes a subset of one or more complete character sets. When the character set is loaded, all characters of the set (up to 64) are loaded into the printer's WCGM; only those characters that are referred to by a translate table can be printed.

TABLE Module Structure

The character arrangement table data following the header information is composed of the following components:

- A 256-byte translate table
- Four 2-byte fields for codes identifying character sets and their WCGM sequence numbers
- Four 4-byte fields for graphic character modification module names

The translate table consists of 256 one-byte entries, each pointing to one of 64 positions within one of four WCGMs:

- Bits 0 and 1 of each translate table byte refer to one of four WCGMs and bits 2 through 7 point to one of 64 addresses (0-63) within the WCGM. If SETPRT loads a character set into a WCGM other than the WCGM called for, SETPRT, using a copy of the translate table, alters bits 0 and 1 of each non-X'FF' byte of the translate table to correspond with the WCGM loaded. Figure 66 on page 176 describes the structure of the character arrangement table module.

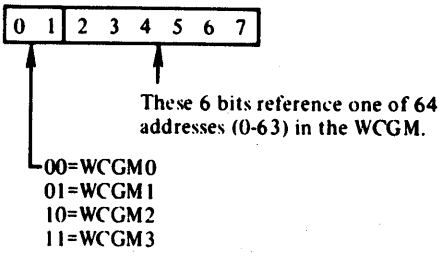


Figure 66. Character Arrangement Table Module Structure

- A byte value of X'FF' indicates an invalid character and prints as a blank and gives a data check. The data check is suppressed if the block data check option is selected.
- One translate table can address multiple WCGMs, and multiple translate tables can address one WCGM. The translate tables supplied by IBM address either one or two WCGMs.

The next two components provide the linkage to character sets and graphic character modification modules. They consist of four 2-byte fields containing character set IDs with their corresponding WCGM sequence numbers, followed by four 4-character names of graphic character modification modules. The format is as follows:

- Each CGMID is a 1-byte character set ID containing two hexadecimal digits that refers to a library character set (as listed in the IBM 3800 Printing Subsystem Programmer's Guide). Each WCGMN0 refers to the corresponding WCGM sequence (X'00' to X'03'). Each name is the 4-character name of a graphic character modification module.

CGMID0	WCGMN00	CGMID1	WCGMN01
CGMID2	WCGMN02	CGMID3	WCGMN03
Name1			
Name2			
Name3			
Name4			

- Most of the standard character arrangement tables do not need graphic character modification. The names are blank (X'40's) if no modules are referred to.
- The CGMID*x* and the WCGMN0*x* are both X'00' when there are no character sets referred to after the first one.

TABLE Module Listing

Figure 67 on page 177 shows the listing of a character arrangement table module. Each of the notes following the figure describes the item in the figure that is marked with the circled number.

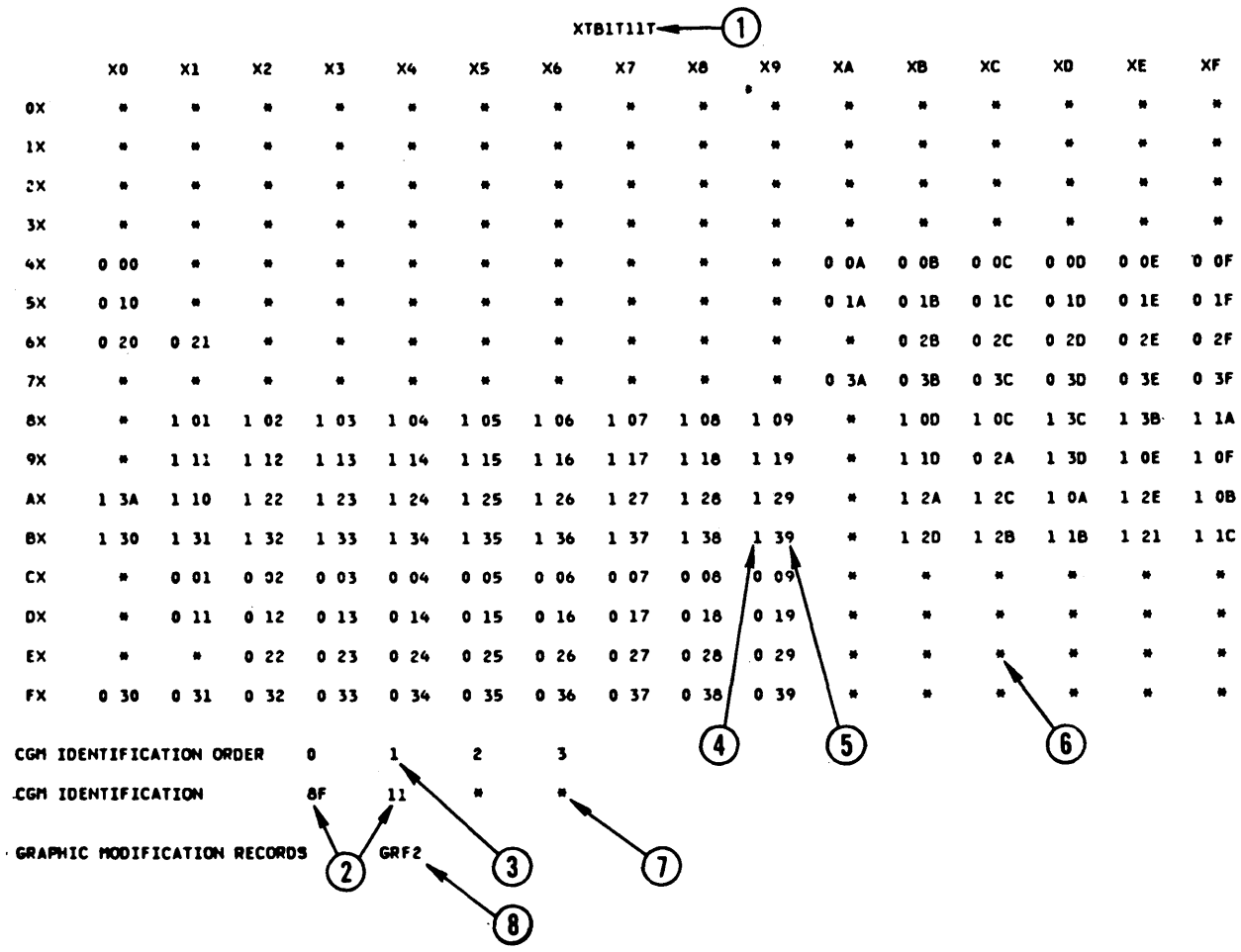


Figure 67. IEBIMAGE Listing of a Character Arrangement Table Module

Notes to Figure 67:

1. The name of the character arrangement table module, as it exists in the image library's directory (including the 4-byte system-assigned prefix).

2. The 1-byte identifier of an IBM-supplied character set (in this example, the Text 1 and Text 2 character sets, whose identifiers are X'8F' and X'11').

   All character sets in SYS1.IMAGELIB or a user-specified image library are represented by odd-numbered identifiers. If the character set identifier specified is even-numbered, it is incremented by one at print time and the character set with that identifier is loaded.

3. The sequence number of the WCGM that is to contain the character set indicated below it (in this example, the second WCGM, whose identifier is 1).

4. The sequence number of the WCGM that contains the scan pattern for the 8-bit data code that locates this translate table entry.

5. Your 8-bit data code X'B9' transmitted to the 3800 Model 3 addresses this, the B9 location in the translate table, where the value X'39' in turn is the index into the WCGM that contains the scan pattern to be used (in this example, the Text 2 superscript 9).

6. Your 8-bit data code X'B9' transmitted to the 3800 addresses this, the B9 location in the translate table, where the value X'39' in turn is the index into the WCGM that contains the scan pattern to be used (in this example, the Text 2 superscript 9).

7. An asterisk is shown in the listing for each translate table entry that contains X'FF'. This indicates that the 8-bit data code that addresses this location does not have a graphic defined for it and is therefore unprintable.

8. An asterisk in the list of character set identifiers indicates that no character set is specified to use the corresponding WCGM. If you specify 7F or FF as a character set identifier (to allow accessing a WCGM without loading it), a 7F or FF prints here.

9. The name of a graphic character modification module, as the name exists in the library's directory (including the system-assigned prefix).

   When you specify a graphic character modification module to be associated with a character arrangement table, you must specify the OPTION statement with the DEVICE parameter (for the 3800 Model 3) to ensure that the system assigns the correct prefix (GRF2) to the graphic character modification module name.

CREATING A GRAPHIC CHARACTER MODIFICATION MODULE

The 3800 graphic character modification module is variable length and contains up to 64 segments. Each segment contains the 1 byte (for the 3800 Model 1) or 6 bytes (for the 3800 Model 3) of descriptive information and the 72-byte (for the 3800 Model 1) or 120-byte (for the 3800 Model 3) scan pattern of a graphic character.

The graphic character modification module is created using the IEBIMAGE program's INCLUDE, GRAPHIC, and NAME utility control statements.

The INCLUDE statement identifies an existing graphic character modification module that is to be copied and used as a basis for the new module.

The OPTION statement with the DEVICE parameter is required to create graphic character modification modules in the 3800 Model 3 compatibility mode module format.

A GRAPHIC statement, when followed by one or more data statements, defines a user-designed character. A GRAPHIC statement can also select a character segment from another graphic character modification module. Each GRAPHIC statement causes a segment to be created for inclusion in the new module.

The NAME statement identifies the new module and indicates that the module is to be added to the library or is to replace an existing module of the same name. More than one GRAPHIC statement can be coded between the INCLUDE and NAME statements, and all such GRAPHIC statements apply to the same graphic character modification module.

GRAPHIC Module Structure

The graphic character modification data following the header information is a series of 73-byte segments for the 3800 Model 1 and 126-byte segments for the 3800 Model 3. A maximum of 64 such segments is allowed in a module. The module structure is shown in Figure 68.

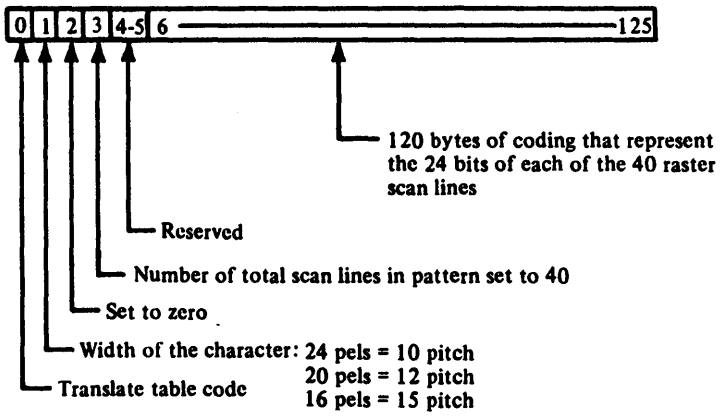


Figure 68. Graphic Character Modification Module Structure

When a graphic character is to be modified, the 3800 uses the translate table code to index into the translate table. The contents found at that location (a 1-byte WCGM code) determine the WCGM location into which the scan pattern and character data are to be placed.

FOR THE 3800 MODEL 1: The 72-byte graphic definition that makes up the scan pattern and system data for one character is divided into 24 3-byte groups. Each 3-byte group represents a horizontal row of 18 1-bit elements (plus parity information).

FOR THE 3800 MODEL 3: The 120-byte graphic definition that makes up the scan pattern for one character is divided into 40 3-byte groups. Each 3-byte group represents a horizontal row of 24 1-bit elements.

GRAPHIC Module Listing

Figure 69 on page 180 shows an extract from a listing of a graphic character modification module. This extract contains the listing of two segments of the module. Each of the notes following the figure describes the item in the figure that is marked with the circled number.

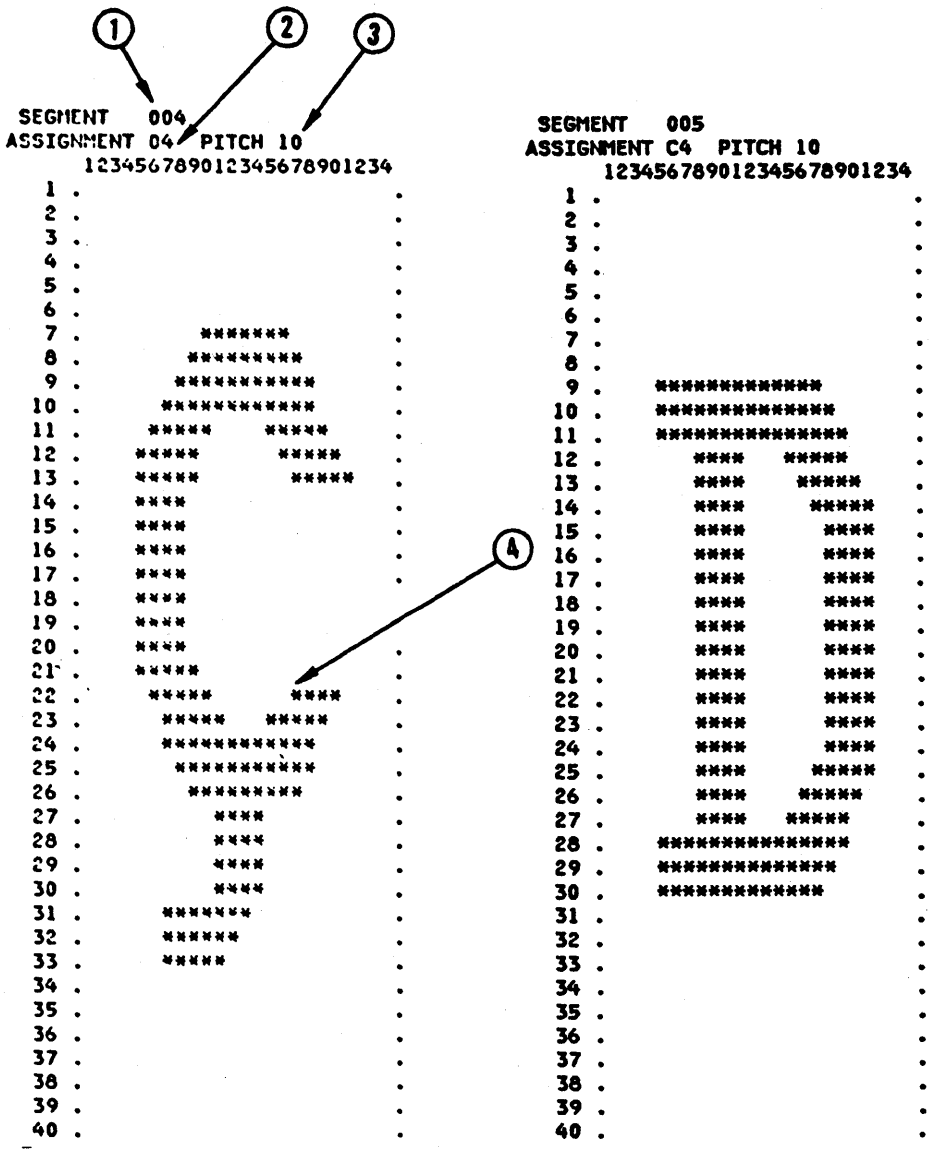


Figure 69. IEBIMAGE Listing of Two Segments of a Graphic Character Modification Module

Notes to Figure 69:

1. The segment number of the character segment within the module.
2. The 8-bit data code for the character.
3. The pitch of the character.
4. The scan pattern for the character. A dollar sign ($) is printed instead of an asterisk if the bit specified is out of the pitch range.

CREATING A LIBRARY CHARACTER SET MODULE

The 3800 library character set module is a fixed-length module made up of 64 segments. Each segment contains the 73 bytes (for the 3800 Model 1) or 126 bytes (for the 3800 Model 3) of information including the scan pattern of a graphic character and a code (00-3F) that identifies the WCGM location into which the scan pattern is to be loaded.

The library character set module is created using the INCLUDE, CHARSET, and NAME control statements.

The INCLUDE statement identifies an existing module.

The OPTION statement with the DEVICE parameter is required to create library character set modules in the 3800 Model 3 compatibility mode module format.

A CHARSET statement, when followed by one or more data statements, defines a user-designed character. A CHARSET statement can also select a character segment from another library character set or from a graphic character modification module.

The NAME statement specifies the ID of the character set being created and indicates if it is to replace an existing module. More than one CHARSET statement can be coded between the INCLUDE and NAME statements; all such CHARSET statements apply to the same library character set module.

CHARSET Module Structure

The library character set data following the header information is a series of 73-byte segments for the 3800 Model 1 and 126-byte segments for the 3800 Model 3. Each module contains 64 segments. For each segment left undefined in a library character set module, IEBIMAGE inserts the graphic symbol for an undefined character. The structure of a library character set module is shown in Figure 70.

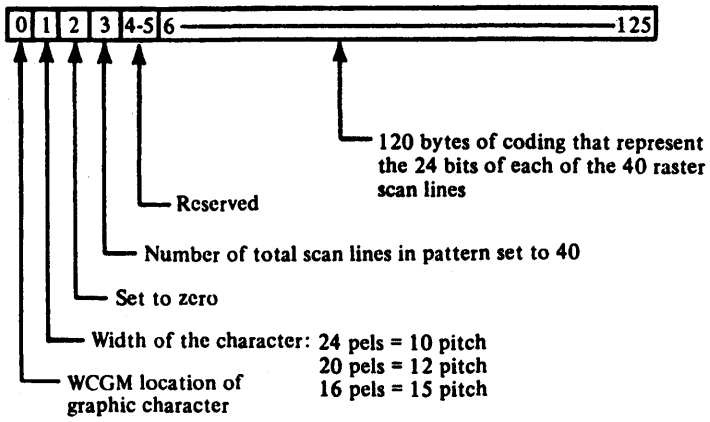


Figure 70. Library Character Set Module Structure

A library character set is loaded directly into a WCGM. SETPRT uses the 6-bit code contained in the first byte of each 73-byte segment (for the 3800 Model 1) or 126-byte segment (for the 3800 Model 3) as the address of the WCGM location into which the remaining 72 bytes (for the 3800 Model 1) or 125 bytes (for the 3800 Model 3) are loaded.

FOR THE 3800 MODEL 1: The 73-byte graphic definition that makes up the scan pattern for one character is divided into 24 3-byte groups. Each 3-byte group represents a horizontal row of 18 1-bit elements.

FOR THE 3800 MODEL 3: The 126-byte graphic definition that makes up the scan pattern for one character is divided into 40 3-byte groups. Each 3-byte group represents a horizontal row of 24 1-bit elements.

CHARSET Module Listing

Figure 71 shows an extract from a listing of a library character set module. This extract contains the listing of two segments of the library character set. The numbered notes that follow the figure describe the items marked with the circled numbers.

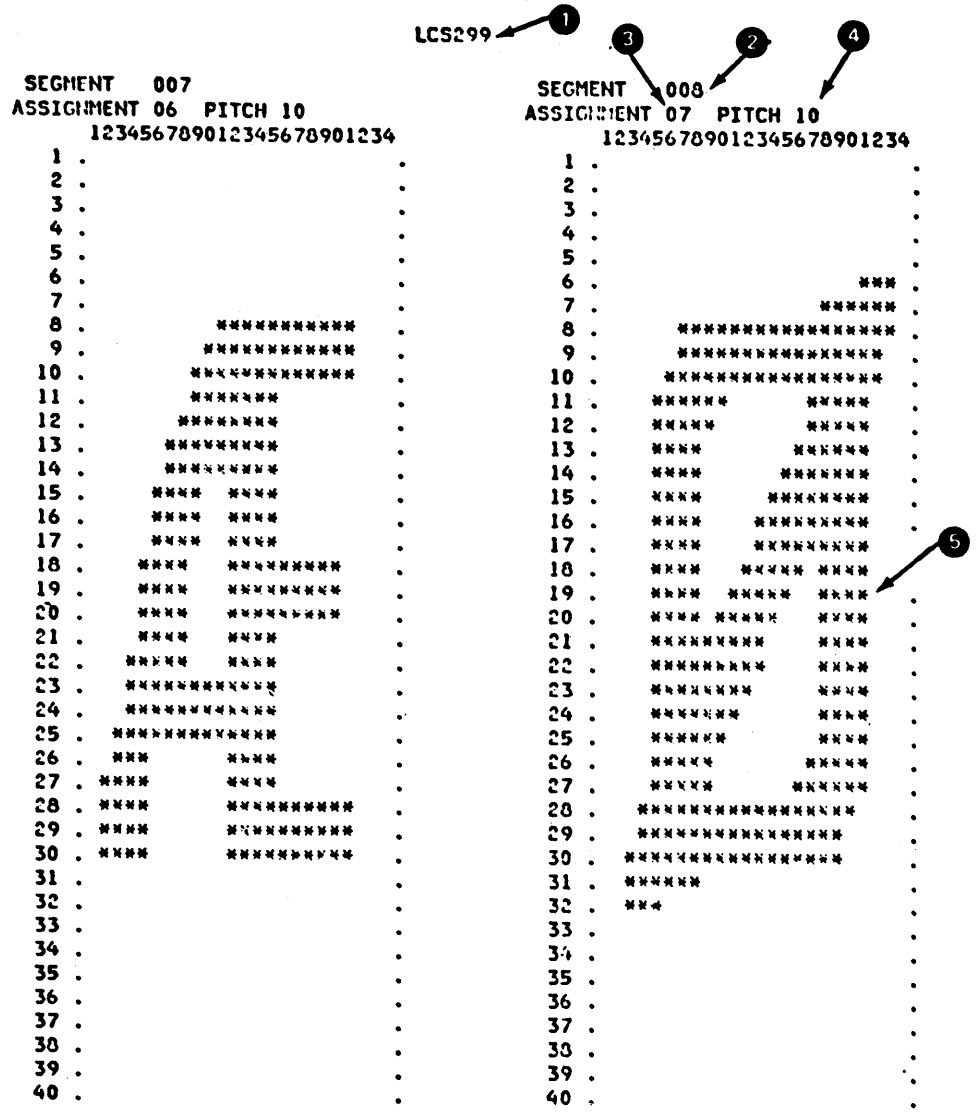


Figure 71. IEBIMAGE Listing of Two Segments of a Library Character Set

Notes to Figure 71:

1. The name of the library character set module, including the 4-byte system-assigned prefix.
2. The segment number of the character segment within the module.
3. The 6-bit code for the WCGM location.
4. The pitch of the character.
5. The scan pattern for the character. A dollar sign ($) is printed instead of an asterisk if the bit specified is out of the pitch range.

INPUT AND OUTPUT

IEBIMAGE uses the following input:

- A control data set that contains utility control statements
- Source statements produced by the Character Conversion Aid

IEBIMAGE produces the following output:

- A new module or modules for use with the 3800 Model 1 and Model 3 printers, 3262 Model 5 printer, or the 4248 printer, to be stored in an image library. These may be of one of the following types:
  - Forms control buffer modules (3800 or 4248)
  - Copy modification modules (3800 only)
  - Character arrangement table modules (3800 only)
  - Graphic character modification modules (3800 only)
  - Library character set modules (3800 only)

  Note that, in building a 4248 FCB module, either a 4248 (prefix FCB4) or a 3211 (prefix FCB2) format FCB may be used. IEBIMAGE prefixes the name with FCB4 first; then if no module exists with that name, the prefix is changed to FCB2.
- An output data set listing for each new module which includes:
  - Module identification
  - Utility control statements used in the job
  - Module contents
  - Messages and return codes

RETURN CODES

IEBIMAGE returns a code in register 15 that represents the most severe error condition encountered during the program execution. This return code is also printed in the output listing. The codes are described Figure 72 on page 184.

Codes	Meaning
00 (00 hex)	Successful completion; operation(s) performed as requested.
04 (04)	Operation(s) performed; investigate messages for exceptional circumstances.
08 (08)	Operation(s) not performed; investigate messages.
12 (0C)	Severe exception; processing may end.
16 (10)	Catastrophic exception; the job step is terminated.
20 (14)	SYSPRINT data set could not be opened; the job step is terminated.
24 (18)	User parameter list invalid; the job step is terminated.

Figure 72. IEBIMAGE Return Codes

CONTROL

IEBIMAGE is controlled by job control statements and utility control statements.

JOB CONTROL STATEMENTS

Figure 73 shows the job control statements for IEBIMAGE.

Statement	Use
JOB	Initiates the job.
EXEC	Specifies the program name (PGM=IEBIMAGE) or, if the job control statements reside in the procedure library, the procedure name. No PARM parameters can be specified.
SYSPRINT DD	Defines the sequential message data set used for listing statements and messages on the system output device.
SYSUT1 DD	Defines the library data set (SYS1.IMAGELIB or a user-defined library).
SYSIN DD	Defines the control data set, which normally resides in the input stream.

Figure 73. Job Control Statements for IEBIMAGE

SYSPRINT DD Statement

Block size for the SYSPRINT data set should be 121 or a multiple of 121. Any blocking factor can be specified. The first character of each 121-byte output record is an ANSI control character.

SYSUT1 DD Statement

To ensure that the library data set is not updated by other jobs while the IEBIMAGE job is running, DISP=OLD should be specified on the SYSUT1 DD statement.

Note that the system will only attempt to locate modules in SYS1.IMAGELIB if the device is a 3800 printer.

SYSIN DD Statement

Block size for the SYSIN data set should be 80 or a multiple of 80. Any blocking factor can be specified.

UTILITY CONTROL STATEMENTS

IEBIMAGE is controlled by the following utility control statements listed in Figure 74.

Continuation requirements for utility control statements are discussed in "Continuing Utility Control Statements" on page 5.

Statement	Use
FCB	Creates a 3800 or 4248 forms control buffer module and stores it in an image library.
COPYMOD	Creates a 3800 copy modification module and stores it in an image library.
TABLE	Creates a 3800 character arrangement table module and stores it in an image library.
GRAPHIC	Creates a 3800 graphic character modification module and stores it in an image library.
CHARSET	Creates a 3800 library character set module and stores it in an image library.
INCLUDE	Identifies an existing image library module to be copied and used as a basis for the new module.
NAME	Specifies the name of a new or existing library module.
OPTION	Specifies optional 3800 Model 3 or 4248 printer compatibility, or COPYMOD overrun lines per inch for an IEBIMAGE job.

Figure 74. Utility Control Statements for IEBIMAGE

Operation Groups

IEBIMAGE utility control statements are grouped together to create or print a library module. Each group of statements is called an **operation group**. Your job's input stream can include many operation groups. The operation groups (shown below without operands) that can be coded are:

- To create or print an FCB module:

```
[OPTION]
[INCLUDE]
FCB
NAME
```

Note: It is not possible to print a 4248 FCB module without coding some valid operation on the FCB statement.

- To create or print a copy modification module:

```
[INCLUDE]
[OPTION]
COPYMOD
[additional COPYMOD statements]
NAME
```

- To create or print a character arrangement table module:

```
[INCLUDE]
[OPTION]
TABLE
NAME
```

- To create or print a graphic character modification module:

```
[INCLUDE]
[OPTION]
{GRAPHIC|GRAPHIC, followed immediately by
data statements}
[additional GRAPHIC statements]
NAME
```

- To create or print a library character set module:

```
[INCLUDE]
[OPTION]
{CHARSET|CHARSET, followed immediately by
data statements}
[additional CHARSET statements]
NAME
```

To print a module, you need only supply the function statement (that is, FCB, COPYMOD, TABLE, GRAPHIC, or CHARSET) with no operands specified, followed by the NAME statement naming the module. However, it is not possible to print a 4248 FCB module without coding some valid operation on the FCB statement.

FCB STATEMENT

The FCB statement specifies the contents of a forms control buffer (FCB) module for the 3800, 3262 Model 5, or 4248 printer: spacing codes (lines per inch), channel codes (simulated carriage-control channel punches), and the size of the form. For the 4248 printer, the FCB statement also specifies print position for the horizontal copy feature and printer speed, and whether the FCB image is to be used as a default.

The FCB statement must always be followed by a NAME statement, and can only be preceded by an INCLUDE statement if DEVICE=4248 is specified on an OPTION statement.

An FCB statement with no operands specified, followed by a NAME statement that identifies a 3800 FCB module in the image library, causes the module to be formatted and printed. 3262 Model 5 and 4248 FCB modules cannot be printed by the FCB statement unless a valid operation is performed on them. To build an FCB module, you code the FCB statement with at least one operand. The format of a printed FCB module is shown in "FCB Module Listing" on page 172.1.

The format of the FCB statement is:

[label]	FCB	[LPI=(([,])[,([,])...])] [,CHx=(line[,line...])[,CHx=(line...)...]] [,SIZE=length] [,LINES=lines] [,COPYP=position] [,PSPEED={L|M|H|N}] [,DEFAULT={YES|NO}]

The COPYP, PSPEED, and DEFAULT parameters are valid only for a 4248 FCB module (not for the 3800 or 3262 Model 5).

COPYMOD STATEMENT

A copy modification module consists of header information followed by one or more modification segments. The header information contains the module's name and length. Each modification segment contains the text to be printed, identifies the copy (or copies) the text applies to, and specifies the position of the text on each page of the copy.

A COPYMOD statement specifies the contents of one of the modification segments of a copy modification module. More than one COPYMOD statement can be coded in an operation group; all COPYMOD statements so coded apply to the same copy modification module.

IEBIMAGE analyzes the modification segments specified for a copy modification module to anticipate line overrun conditions that might occur when the module is used in the printer. A line overrun condition occurs when the modification of a line is not completed in time to print that line. The time available for copy modification varies with the vertical line spacing (lines per inch) at which the printer is being operated.

When IEBIMAGE builds a copy modification module from the user's specifications, the program calculates an estimate of the time the modification will require during the planned printing. If the modification can be done in the time available for printing a line at 12 LPI (lines per inch), it can also be done at 6 or 8 LPI (for the Model 1), or 6, 8, or 10 LPI (for the Model 3). (Note that 6, 8, 10, and 12 LPI are the only print densities available on the 3800 Model 3 printer.) However, if the copy modification module being built is too complex to be done in the time available for printing a line at 6 LPI, it certainly cannot be done at 8, 10 (for the Model 3 only), or 12 LPI. (Note that at 10 and 12 LPI there is much less time available for printing a line than at 6 LPI.)

When IEBIMAGE determines that a copy modification module is likely to cause an overrun if it is used when printing at a specified number of lines per inch, the program produces a warning message to that effect. If the warning applies to 6 LPI, the overrun condition is also applicable to 8, 10 (for the Model 3 only), and 12 LPI. If the warning applies to 8 LPI, the condition is also applicable for 10 (for the Model 3 only) and 12 LPI. If the warning applies to 10 LPI, the condition also applies to 12 LPI.

If you are planning to use a particular copy modification module only while printing at 6 LPI, you can request suppression of the unwanted warning messages for 8, 10 (for the Model 3 only), and 12 LPI by specifying the OPTION statement with 6 as the value of

the OVERRUN parameter. If you are planning to print only at 8 LPI, you can use the OPTION statement with OVERRUN = 8 to request suppression of the unwanted warning messages for 10 (for the Model 3 only) and 12 LPI. For more information on coding OVERRUN, see "Using OVERRUN" on page 192.

For information about using your copy modification module, see IBM 3800 Printing Subsystem Programmer's Guide. The copy modification text can be printed using the same character size or style, or one different from the size or style used to print the data in the output data set.

The COPYMOD statement must always be followed by a NAME statement or another COPYMOD statement and can be preceded by an INCLUDE statement. When more than one COPYMOD statement is coded, IEBIMAGE sorts the statements into order by line number within copy number. A COPYMOD statement with no operands specified, followed by a NAME statement that identifies a copy modification module, is used to format and print the module. The format of the printed module is shown under "COPYMOD Module Listing" on page 174.

The format of the COPYMOD statement, when used to create a copy modification module's segment, is:

[label]	COPYMOD	COPIES=(starting-copy[,copies]), LINES=(starting-line[,lines]), POS=position, TEXT=(([d]t,'text')[,([d]t,'text')

TABLE STATEMENT

The TABLE statement is used to build a character arrangement table module. When a character arrangement table is built by IEBIMAGE and an INCLUDE statement is specified, the contents of the copied character arrangement table are used as a basis for the new character arrangement table. If an INCLUDE statement is not specified, each translate table entry in the new character arrangement table module is initialized to X'FF', the graphic character modification module name fields are set with blanks (X'40'), and the first character set identifier is set to X'83' (which is the Gothic 10-pitch set). The remaining identifiers are set to X'00'.

After the character arrangement table is initialized, IEBIMAGE modifies the table with data specified in the TABLE statement: character set identifiers, names of graphic character modification modules, and specified translate table entries. The character arrangement table, when built, must contain a reference to at least one printable character. Only one TABLE statement can be specified for each operation group. The TABLE statement can be preceded by an INCLUDE statement and an OPTION statement and must always be followed by a NAME statement.

A TABLE statement with no operands specified, followed by a NAME statement that identifies a character arrangement table module in the library, causes the module to be formatted and printed. The TABLE statement should be preceded by an OPTION statement with the DEVICE=3800M3 parameter for a 3800 Model 3. The format of the printed character arrangement table module is shown under "TABLE Module Listing" on page 176.

The format of the TABLE statement is:

[label]	TABLE	[CGMID=(set0[,set1...])] [,GCMLIST={(gcm1[,gcm2..])|DELETE}] [,LOC=((xloc[,(cloc[,setno]|FF)])]

GRAPHIC STATEMENT

The GRAPHIC statement specifies the contents of one or more of the character segments of a graphic character modification module. A graphic character modification module consists of header information followed by from 1 to 64 character segments. Each character segment contains

- The character's 8-bit data code, its scan pattern, and its pitch (for the 3800 Model 1)
- Six bytes of descriptive information and the 120-byte scan pattern (for the 3800 Model 3)

By using the INCLUDE statement, you can copy an entire module, minus any segments deleted using the DELSEG keyword. In addition, you can select character segments from any module named with the GCM keyword on the GRAPHIC statement. The GRAPHIC statement can also specify the scan pattern and characteristics for a new character.

The GRAPHIC statement must always be followed by a NAME statement, another GRAPHIC statement, or one or more data statements. The OPTION statement with the DEVICE parameter must precede the GRAPHIC statement to create a graphic character modification module in the 3800 Model 3 compatibility mode module format. The GRAPHIC statement can be preceded by an INCLUDE statement. More than one GRAPHIC statement can be coded in the operation group. The operation group can include GRAPHIC statements that select characters from existing modules and GRAPHIC statements that create new characters. The GRAPHIC statement, preceded by an INCLUDE statement, can be used to delete one or more segments from the copy of an existing module to create a new module.

A GRAPHIC statement with no operands specified, followed by a NAME statement that identifies a graphic character modification module, is used to format and print the module. When you specify a graphic character modification module to be printed for a 3800 Model 3, you must specify the OPTION statement with the DEVICE parameter to ensure that the system assigns the correct prefix (GRF2) to the graphic character modification module name.

The format of the GRAPHIC statement, when it is used to select a character segment from another graphic character modification module, is:

[label]	GRAPHIC	[REF=((segno[,xloc])[,(segno[,xloc])..]) [,GCM=name]

The format of the GRAPHIC statement, when it is used to specify the scan pattern and characteristics of a newly-created character, is:

[label]	GRAPHIC	ASSIGN=(xloc[,pitch]) data statements

CHARSET STATEMENT

The CHARSET statement specifies the contents of one or more of the character segments of a library character set module. A library character set module consists of header information followed by 64 character segments. Each character segment contains the character's 6-bit code for a WCGM location, its scan pattern, and its pitch. You can use the INCLUDE statement to copy an entire module, minus any segments deleted using the DELSEG keyword. In addition, you can use the CHARSET statement to select character segments from any module named with a library character set ID or the GCM keyword. The CHARSET statement can also specify the scan pattern and characteristics for a new character.

The CHARSET statement must always be followed by a NAME statement, another CHARSET statement, or one or more data statements. The CHARSET statement must be preceded by an OPTION statement with the DEVICE parameter to create library character set modules in the 3800 Model 3 compatibility mode module format. The CHARSET statement can be preceded by an INCLUDE statement. More than one CHARSET statement can be coded in the operation group. The operation group can include CHARSET statements that select characters from existing modules and CHARSET statements that create new characters. The CHARSET statement, preceded by an INCLUDE statement, can be used to delete one or more segments from the copy of an existing module to create a new module.

A CHARSET statement with no operands specified, followed by a NAME statement that identifies a library character set module, is used to format and print the module.

The format of the CHARSET statement, when it is used to select a character segment from another module, is:

[label]	CHARSET	[REF=((segno,cloc)[,(segno,cloc)...]) [,{GCM=name|ID=xx}]]

The format of the CHARSET statement, when it is used to specify the scan pattern and characteristics of a newly-created character, is:

[label]	CHARSET	ASSIGN=(cloc[,pitch]) data statements

INCLUDE STATEMENT

When an IEBIMAGE operation group is used to create a new module, the INCLUDE statement can identify an existing image library module to be copied and used as a basis for the new module. When the operation group is used to update an image library module, the INCLUDE statement identifies the module to be referred to and must be specified.

The format of the INCLUDE statement is:

[label]	INCLUDE	module name [,DELSEG=(segno[,segno...])]

- When the INCLUDE statement is coded in an operation group, it must precede any FCB, COPYMOD, TABLE, GRAPHIC, or CHARSET statements.
- Only one INCLUDE statement should be coded for each operation group. If more than one is coded, only the last is used; the others are ignored.
- You can code an INCLUDE statement for an FCB module only if the DEVICE=4248 parameter is specified on the OPTION statement. Either 3211 format or 4248 format FCBs may be included. IEBIMAGE attempts to locate the 4248 format FCB first; if it is not found, IEBIMAGE looks for the 3211 format.
- You cannot copy a 3800 FCB module with INCLUDE.

NAME STATEMENT

The NAME statement can name a new library module to be built by the IEBIMAGE program. The NAME statement can also specify the name of an existing library module. The NAME statement is required, and must be the last statement in each operation group.

The format of the NAME statement is:

[label]	NAME	module name[(R)]

OPTION STATEMENT

The OPTION statement with the DEVICE=3800M3 parameter is required to create library character set modules and graphic character modification modules in a form usable on the 3800 Model 3. The OPTION statement with the DEVICE=3800M3 parameter is optional when creating copy modification modules and character arrangement table modules.

The OPTION statement with the DEVICE=4248 parameter is required to create a forms control buffer module for the 3262 Model 5 or 4248 printer. DEVICE=4248 cannot be used to create any module other than an FCB.

The OPTION statement with the OVERRUN parameter is used only in a COPYMOD operation group and can be placed before or after any INCLUDE statement for the group. The value in the OVERRUN parameter specifies the greatest line density for which the user wants the overrun warning message IEBA33I to be printed. See "Using OVERRUN" on page 192 for information about overrun conditions and suppression of overrun warning messages.

The format of the OPTION statement is:

[label]	OPTION	[OVERRUN={0|6|8|10|12}] [DEVICE={3800M3|4248}]

When two parameters are specified, they may be listed in any order and separated by a comma.

An effective use of the OPTION statement with the OVERRUN parameter would be to determine the greatest print-line-density (6, 8, 10, 12) at which the copy modification module will be used, then specify that density in the OVERRUN parameter to eliminate the warning messages for higher line densities.

The OPTION statement applies only to the operation group that follows it. If used, the OPTION statement must be specified for each operation group in the job input stream.

Using OVERRUN

Figure 75 shows the listing of segments of a copy modification module where an overrun warning was in order. Even if the OPTION statement specifies OVERRUN=0 and the overrun warning message is not printed, a note is printed to the left of each segment description for which an overrun is possible.

Notes	Segment	Initial Copy Number	Number of Copies	Initial Line Number	Number of Lines	Initial Print Position	Number of Characters
	1	1	200	10	96	10	180
Note(0)[1]	2	2	200	10	96	11	180
Note(1)[2]	3	3	200	10	96	12	180
Note(2)[3]	4	4	200	10	96	10	180
Note(2)	5	5	200	10	96	11	180
Note(3)[4]	6	6	200	10	96	12	180
Note(3)	7	7	200	10	96	10	180
Note(3)	8	8	200	10	96	11	180
Note(3)	9	9	200	10	96	12	180

Figure 75. IEBIMAGE Listing of a Copy Modification Module with Overrun Notes

Notes to Figure 75:

1. Note 0 indicates that, for segment 1, you might have a copy modification overrun if your are printing at 12 LPI.
2. Note 1 indicates that, for segments 2 and 3, you might have a copy modification overrun if you are printing at 8 LPI.
3. Note 2 indicates that, for segments 4 and 5, you might have a copy modification overrun if you are printing at 8 or 12 LPI.
4. Note 3 indicates that, for segments 6, 7, 8, and 9, you might have a copy modification overrun if you are printing at 6, 8, or 12 LPI. In other words, you might have an overrun at any LPI.

Factors used in determining a line overrun condition are:

- Number of modifications per line
- Number of segments per module

Combining COPYMOD segments reduces the possibility of a line overrun condition.

For the algorithm for calculating when a copy modification module might cause a line overrun condition, see Reference Manual for the IBM 3800 Printing Subsystem.

Parameters	Applicable Control Statements	Description of Parameters
ASSIGN	CHARSET	ASSIGN=(cloc[,pitch]) identifies a newly-created character and its characteristics. The ASSIGN parameter specifies the new character's 6-bit code and its pitch. When IEBIMAGE detects the ASSIGN parameter, the program assumes that all following statements, until a statement without the characters SEQ= in columns 25 through 28 is encountered, are data statements that specify the character's scan pattern. cloc specifies the character's 6-bit code for a WCGM location and can be any value between X'00' and X'3F'. cloc is required when ASSIGN is coded. pitch specifies the character's horizontal size and is one of the following decimal numbers: 10, 12, or 15. If pitch is not specified, the default is 10. At least one data statement must follow a CHARSET statement containing the ASSIGN parameter.

Parameters	Applicable Control Statements	Description of Parameters
ASSIGN	GRAPHIC	**ASSIGN=(xloc[,pitch])** identifies a newly-created character and its characteristics. The ASSIGN parameter specifies the new character's 8-bit data code and its pitch. When IEBIMAGE detects the ASSIGN parameter, it assumes that all following statements, until a statement without the characters SEQ= in columns 25 through 28 is encountered, are data statements that specify the character's scan pattern. xloc specifies the character's 8-bit data code, and can be any value between X'00' and X'FF'. You should ensure that xloc identifies a translate table entry that points to a character position in a WCGM (that is, the translate table entry doesn't contain X'FF'). The xloc is required when ASSIGN is coded. pitch specifies the character's horizontal size and is one of the decimal numbers 10, 12, or 15. If pitch is not specified, the default is 10. At least one data statement must follow a GRAPHIC statement containing the ASSIGN parameter.
CGMID	TABLE	**CGMID=(set0[,set1...])** identifies the character sets that are to be used with the character arrangement table. (The IBM-supplied character sets are described in IBM 3800 Printing Subsystem Programmer's Guide.) When CGMID is specified, all character set identifiers are changed. If only one character set is specified, the other three identifiers are set to X'00'. setx is a 1-byte identifier of a character set. Up to four character set identifiers can be specified; set0 identifies the character set that is to be loaded into the first writable character generation module (WCGM); set1 is loaded into the second WCGM; etc. You should ensure that the character set identifiers are specified in the proper sequence, so that they are coordinated with the translate table entries. For the character set identifiers, see IBM 3800 Printing Subsystem Programmer's Guide.

Parameters	Applicable Control Statements	Description of Parameters
CH*x*	FCB	CH*x*=([*line*[,*line*...]]) specifies the channel code (or codes) and the line number (or numbers) to be skipped to when that code is specified. CH*x* specifies a channel code, where *x* is a decimal integer from 1 to 12. *line* specifies the line number of the print line to be skipped to, and is expressed as a decimal integer. The first printable line on the page is line number 1. The value of *line* cannot be larger than the line number of the last printable line on the form. Only one channel code can be specified for a print line. However, more than one print line can contain the same channel code. **Conventions:** • Channel 1 is used to identify the first printable line on the form. The job entry subsystem and the CLOSE routines for direct allocation to the 3800 with BSAM or QSAM require a channel 1 code even when the data being printed contains no skip to channel 1. • Channel 9 is used to identify a special line. To avoid I/O interrupts that are caused by use of channel 9, count lines to determine the line position. • Channel 12 is used to identify the last print line on the form to be used. To avoid I/O interrupts that are caused by use of channel 12, count lines to determine the page size. • Use of an FCB that lacks a channel code to terminate a skip operation causes a data check at the printer when the corresponding skip is issued. This data check cannot be blocked. If INCLUDE is specified, values for CH*x* may be taken from the included FCB module. See the discussion under *module name*.

Parameters	Applicable Control Statements	Description of Parameters
COPIES	COPYMOD	**COPIES=(**starting-copy**[,**copies**])** specifies the starting copy number and the total number of copies to be modified. starting-copy specifies the starting copy number and is expressed as a decimal integer from 1 to 255. The starting-copy value is required. copies specifies the number of copies that are to contain the modifying text and is expressed as a decimal integer from 1 to 255. When copies is not specified, the default is 1 copy. The sum of starting-copy and copies cannot exceed 256 (255 for JES3).
COPYP	FCB	**COPYP=**position specifies the position (the number of character spaces from the left margin) at which the horizontal copy is to begin printing. position is a decimal number from 2 to 168 which indicates where the horizontal copy printing will start. If your 4248 printer has only 132 print positions, the maximum number you should specify here is 132. If COPYP=0 is coded, any COPYP value previously set in an included FCB module is overridden, and the horizontal copy feature is turned off. You may not specify COPYP=1. If INCLUDE is specified, and the included FCB module is formatted for a 4248 printer only, the default is the COPYP value for the included FCB module. Otherwise, if no COPYP value is specified, the default value is 0. COPYP is not valid for 3800 FCB modules. COPYP is ignored for 3262 Model 5 FCB modules. The COPYP value specified affects the maximum amount of data that may be sent to the printer. Channel programs that are executed with the horizontal copy feature activated **must** set the suppress incorrect length (SLI) bit and have a data length that does not exceed the size of either one half the number of print positions or the smaller of the two copy areas.

Parameters	Applicable Control Statements	Description of Parameters
data statements	GRAPHIC CHARSET	data statements describe the design of the character as it is represented on a character design form. For details of how to design a character and how to use the character design form, see IBM 3800 Printing Subsystem Programmer's Guide. Each data statement represents a line on the design form. Each nonblank line on the design form must be represented with a data statement; a blank line can also be represented with a data statement. You can code up to 24 (for 3800 Model 1) or 40 (for 3800 Model 3) data statements to describe the new character's pattern. On each statement, columns 1 through 18 (for Model 1) or 24 (for Model 3) can contain nonblank grid positions when the character is 10-pitch. Any nonblank character can be punched in each column that represents a nonblank grid position. Columns 1 through 15 (for Model 1) or 20 (for Model 3) can contain nonblank grid positions when the character is 12-pitch. Columns 1 through 15 (for Model 1) or 1 through 16 (for Model 3) can contain nonblank grid positions when the character is 15-pitch.
DEFAULT	FCB	DEFAULT={YES|NO} specifies whether this 4248 FCB image is to be treated as the default image by OPEN processing. Default images are used by the system for jobs that do not request a specific image. If a job does not request a specific FCB image, and the current image is not a default, the operator will be prompted for an FCB image at OPEN time. If INCLUDE is used to copy a 4248 FCB module that was originally specified as a default image, the new module will also be considered a default image unless DEFAULT=NO is now specified. DEFAULT is not valid for 3800 FCB images.
DELETE	TABLE	DELETE specifies that all graphic character modification module name fields are to be set to blanks.

Parameters	Applicable Control Statements	Description of Parameters
DELSEG	INCLUDE	**DELSEG=(**segno[,segno...]**)** specifies the segments of the copied module that are to be deleted when the module is copied. Segment numbers can be specified in any order. In this parameter, segment 1 is used to refer to the first segment of the module. When you code the DELSEG parameter, you should use a current listing of the module's contents to ensure that you are correctly identifying the unwanted segments. You can code the DELSEG parameter only when the named module is a copy modification module, a graphic character modification module, or a library character set module.
DEVICE	OPTION	**DEVICE={3800M3|4248}** specifies printer compatibility mode module formats and processing considerations. **3800M3** specifies 3800 Model 3 compatibility. **4248** specifies that the module created or modified with the FCB statement should be formatted for the 3262 Model 5 or 4248 printer. See Figure 62.1 on page 172 for the format of the 4248 FCB module. If the DEVICE parameter is omitted, modules are created for the 3800 Model 1.
GCM	CHARSET GRAPHIC	**GCM=**name can be coded when the REF parameter is coded and identifies the graphic character modification module that contains the character segments referred to by the REF parameter. name specifies the 1- to 4-character user-specified name of the graphic character modification module. If GCM is coded, REF must also be coded. GCM should not be coded with ID. When neither GCM nor ID is coded, the segments are copied from the IBM-supplied World Trade National Use Graphics graphic character modification module.

Parameters	Applicable Control Statements	Description of Parameters
GCMLIST	TABLE	**GCMLIST=(gcm1[,gcm2...])** names up to four graphic character modification modules to be associated with the character arrangement table. When GCMLIST is specified, all graphic character modification module name fields are changed (if only one module name is specified, the other three name fields are set to blanks). *gcmx* is the 1- to 4-character name of the graphic character modification module. Up to four module names can be specified. The name is put into the character arrangement table, whether or not a graphic character modification module currently exists with that name. However, if the module doesn't exist, IEBIMAGE issues a warning message to the user. The character arrangement table should not be used unless all graphic character modification modules it refers to are stored in an image library.
ID	CHARSET	**ID=*xx*** can be coded when the REF parameter is coded and identifies a library character set that contains the character segments referred to by the REF parameter. *xx* specifies the 2-hexadecimal-digit ID of the library character set module. The second digit must be odd, and '7F' and 'FF' are not allowed. ID should not be coded with GCM. When neither ID nor GCM has been coded, the segments are copied from the IBM-supplied World Trade National Use Graphics graphic character modification module.

Parameters	Applicable Control Statements	Description of Parameters
LINES	COPYMOD	LINES=(starting-line[,lines]) specifies the starting line number, and the total number of lines to be modified. starting-line specifies the starting line number, and is expressed as a decimal integer from 1 to 132. The starting-line value is required. lines specifies the number of lines that are to contain the modification segment's text, and is expressed as a decimal integer from 1 to 132. When lines is not specified, the default is 1 line. The sum of starting-line and lines cannot exceed 133. If the sum exceeds the number of lines specified for the form size (see the "FCB Statement"), the modifying text is not printed on lines past the end of the form.
LINES	FCB	LINES=lines specifies the total number of lines to be contained in an FCB module. lines is the decimal number, from 1 to 256, which indicates the number of lines on the page. When the LINES, SIZE, and LPI parameters are specified in the FCB statement, each parameter value is checked against the others to ensure that there are no conflicting page-length specifications. When LINES is not specified, the form length defaults to the value of LPI multiplied by the value of SIZE, in inches. If no SIZE parameter is specified, LINES defaults to 11 times the value of LPI. If INCLUDE is specified, the value for LINES may be taken from the included FCB module. See the discussion under module name.

Parameters	Applicable Control Statements	Description of Parameters
LOC	TABLE	LOC=((xloc[,{cloc[,setno]|FF}]) [,(xloc...)...]) specifies values for some or all of the 256 translate table entries. Each translate table entry identifies one of 64 character positions within one of the WCGMs. xloc is an index into the translate table, and is specified as a hexadecimal value from X'00' to X'FF'; xloc identifies a translate table entry, not the contents of the entry. cloc identifies one of the 64 character positions within a WCGM, and is specified as a hexadecimal value between X'00' and X'3F'. cloc and setno specify the contents of the translate table entry located by xloc. When cloc is not specified, the default is X'FF', an invalid character. You can specify the same cloc and setno values for more than one xloc. setno identifies one of the WCGMs, and is specified as a decimal integer from 0 to 3. cloc and setno specify the contents of the translate table entry located by xloc. When setno is not specified, the default is 0. The setno cannot be specified unless cloc is also specified. You can specify the same cloc and setno values for more than one xloc.

Parameters	Applicable Control Statements	Description of Parameters
LPI	FCB	**LPI=((*l*[,*n*])[,(*l*[,*n*])...])** specifies the number of lines per inch and the number of lines to be printed at that line spacing. *l* specifies the number of lines per inch, and can be 6, 8, or 12 (for the 3800 Model 1); 6 or 8 (for the 3262 Model 5 or 4248); or 6, 8, 10, or 12 (for the 3800 Model 3). *n* specifies the number of lines at a line spacing of *l*. When the printer uses common-use paper sizes, *n* is a decimal value from 1 to 60 when *l* is 6; from 1 to 80 when *l* is 8; from 1 to 100 when *l* is 10; and from 1 to 120 when *l* is 12. When the printer uses ISO paper sizes, *n* is a value from 1 to 66 when *l* is 6; from 1 to 88 when *l* is 8; from 1 to 110 when *l* is 10; or from 1 to 132 when *l* is 12. For the paper sizes, see IBM 3800 Printing Subsystem Programmer's Guide. It is the user's responsibility to ensure that the total number of lines specified results in a length that is a multiple of 1/2 inch. The total number of lines cannot result in a value that exceeds the usable length of the form. For the 3800, do not specify coding for the top and bottom 1/2 inch of the form; IEBIMAGE does this for you. When the SIZE, LINES, and LPI parameters are specified in the FCB statement, each parameter value is checked against the others to ensure that there are no conflicting page-length specifications. For example, SIZE=35 specifies a 3-1/2 inch length; acceptable LPI values for the 3800 cannot define more than the printable 2-1/2 inches of this length. When you specify more than one (*l*,*n*) pair, *l* must be specified for each pair and *n* must be specified for each pair except the last. When you specify 12 lines per inch, use one of the condensed character sets. If other character sets are printed at 12 lines per inch, the tops or bottoms of the characters may not print.

Parameters	Applicable Control Statements	Description of Parameters
LPI (continued)	FCB	When only *l* is specified, or when *l* is the last parameter in the LPI list, all remaining lines on the page are at *l* lines per inch. When LPI is not specified, all lines on the page are at 6 lines per inch. If the total number of lines specified is less than the maximum number that can be specified, the remaining lines default to 6 lines per inch. If INCLUDE is specified, the value for LPI may be taken from the included FCB module. See the discussion under *module name*.
module name	INCLUDE NAME	*module name* names or identifies a library module. The module name is 1 to 4 alphameric and national ($, #, and @) characters, in any order, or, for a library character set module, a 2-character ID that represents two hexadecimal digits (0-9, A-F), the second digit being odd. Note that 7F and FF cannot be used. For a 3800 INCLUDE operation, the named module must be the same type as the module being created. However, for the 4248 printer, if the named FCB module is not found to exist with the prefix FCB4, an existing 3211 FCB module (prefix FCB2) with the same module name will be used. In this case, the values specified for the LINES, SIZE, CH*x*, and LPI parameters on the FCB statement will default to the values previously specified in the included module if the new values are not compatible with the 3211 printer. If the 3211 module was a default image, the 4248 module will also be a default image unless the DEFAULT parameter is specified as NO.

Parameters	Applicable Control Statements	Description of Parameters
OVERRUN	OPTION	**OVERRUN={0|6|8|10|12}** specifies the greatest number of lines per inch for which message IEBA33I is to be printed for a COPYMOD operation. For example, OVERRUN=8 allows the message for 6 and 8 lines per inch, but suppresses it for 10 and 12 lines per inch. Specifying OVERRUN=0 suppresses message IEBA33I for every case. If you specify OVERRUN=12, none will be suppressed. OVERRUN=10 is valid only for the 3800 Model 3. If the OPTION statement is omitted, the OVERRUN parameter default value is 12, and messages are not suppressed. If the OVERRUN parameter is omitted, the default value is also 12. If the parameter specification is invalid (for instance, if OVERRUN=16 is specified), the entire operation group does not complete successfully. For details of using the OVERRUN parameter with COPYMOD, see "Using OVERRUN" on page 192.
POS	COPYMOD	**POS=position** specifies the starting print position (the number of character positions from the left margin) of the modifying text. position specifies the starting print position and is expressed as an integer from 1 to 204. See the restriction noted for the TEXT parameter below. The maximum number of characters that can fit in a print line depends on the pitch of each character and the width of the form. For the maximum number of characters that can fit in a print line for each form width, see IBM 3800 Printing Subsystem Programmer's Guide.

Parameters	Applicable Control Statements	Description of Parameters
PSPEED	FCB	**PSPEED={L|M|H|N}** specifies the print speed for the 4248 printer. Note that printer speed affects the quality of printing; LOW speed provides the best quality. **L or LOW** sets the printer speed to 2200 lines per minute (LPM). **M or MEDIUM** sets the printer speed to 3000 LPM. **H or HIGH** sets the printer speed to 3600 LPM. **N or NOCHANGE** indicates that the current printer speed should remain unchanged. If INCLUDE is specified, and the included module is formatted for a 4248 printer only, the default is the PSPEED value for the included FCB module. Otherwise, the default is NOCHANGE (or N). PSPEED is not valid for 3800 FCB modules. PSPEED is ignored for 3262 Model 5 FCB modules.
(R)	NAME	**(R)** indicates that this module is to be replaced by a new module with the same name, if it exists. (R) must be coded in parentheses.

Parameters	Applicable Control Statements	Description of Parameters
REF	CHARSET	REF=((segno,cloc)[,(segno,cloc)...]) identifies one or more character segments within an existing graphic character modification module or library character set module. If the reference is to a GCM, the scan pattern and pitch of the character referred to are used, and a 6-bit WCGM location code is assigned. If the reference is to a character in a library character set, the entire segment, including the 6-bit WCGM location code, is used, unless the cloc subparameter is specified for that segment. The REF parameter cannot be used to change a character's pitch or scan pattern. segno is the segment number, a decimal integer between 1 and 999. When a character segment is copied from the IBM-supplied World Trade National Use Graphics graphic character modification module, segno can be greater than 64. When the character segment is copied from a graphic character modification or library character set module built with the IEBIMAGE program, segno is a number from 1 to 64. cloc specifies a 6-bit code that points to a WCGM location, and can be any value between X'00' and X'3F'. When a library character set segment is referred to, if cloc is not specified, the character's 6-bit code remains unchanged when the segment is copied. If a graphic character modification segment is referred to, cloc must be specified. The REF parameter can be coded in a CHARSET statement that includes the ASSIGN parameter.

Parameters	Applicable Control Statements	Description of Parameters
REF	GRAPHIC	REF=([segno[,xloc]][,[segno[,xloc]]...]) identifies one or more character segments within an existing graphic character modification module. Each character segment contains the scan pattern for a character and the 6 bytes of descriptive information (used to locate its translate table entry). The 6 bytes of descriptive information can be respecified with the xloc subparameter. The REF parameter cannot be used to change a character's pitch or scan pattern. segno is the segment number, a decimal integer between 1 and 999. When a character segment is copied from the IBM-supplied World Trade National Use Graphics graphic character modification module, segno can be greater than 64. When the character segment is copied from a graphic character modification module built with the IEBIMAGE program, segno is a number from 1 to 64. xloc specifies an 8-bit data code for the character, and can be any value between X'00' and X'FF'. You should ensure that xloc identifies a translate table entry that points to a character position in the WCGM (that is, the translate table entry doesn't contain X'FF'). If xloc is not specified, the character's 8-bit data code remains unchanged when the segment is copied. The REF parameter can be coded in a GRAPHIC statement that includes the ASSIGN parameter.
SEQ	CHARSET GRAPHIC	SEQ=nn specifies the sequence number that must appear in columns 25 through 30 of the data statement and identifies the line as a data statement; nn specifies a line number (corresponding to a line on the character design form) and is a 2-digit decimal number from 01 to 40.

Parameters	Applicable Control Statements	Description of Parameters
SIZE	FCB	**SIZE=**length specifies the vertical length of the form, in 10ths of an inch. See IBM 3800 Printing Subsystem Programmer's Guide for the allowable lengths for the 3800. The complete length of the form is specified (for example, with the 3800, SIZE=110 for an 11-inch form) even though the amount of space available for printing is reduced by the 1/2-inch top and bottom areas where no printing occurs. When the SIZE, LINES, and LPI keywords are specified in the FCB statement, each parameter value is checked against the others to ensure that there are no conflicting page-length specifications. For example, SIZE=35 specifies a 3-1/2 inch length; acceptable LPI values for the 3800 cannot define more than the printable 2-1/2 inches of this length. When SIZE is not specified, the form length defaults to the value specified in LINES. If LINES is not specified, SIZE is assumed to be 11 inches (110). If INCLUDE is specified, the value for SIZE may be taken from the included FCB module. See the discussion under module name.

Parameters	Applicable Control Statements	Description of Parameters
TEXT	COPYMOD	TEXT=(([d]t,'text')[,([d]t,'text')...]) specifies the modifying text. The text is positioned on the form based on the LINES and POS parameters and replaces the output data set's text in those positions. d specifies a duplication factor (that is, the number of times the text is to be repeated). The d is expressed as a decimal integer from 1 to 204. If d is not specified, the default is 1. t specifies the form in which the text is entered: C for character, or X for hexadecimal. The t is required. text specifies the text and is enclosed in single quotation marks. If the text type is C, you can specify any valid character. Blanks are valid characters. A single quotation mark is coded as two single quotation marks. You are not allowed to specify a character that results in a X'FF'. If the text type is X, the text is coded in increments of two characters that specify values between X'00' and X'FE'. You are not allowed to specify X'FF'. The sum of the starting print position (see the POS parameter) and the total number of text characters cannot exceed 205. If the width of the form is less than the amount of space required for the text (based on character pitch, starting position, and number of characters), characters are not printed past the right margin of the form. If a text character specifies a character whose translate table entry contains X'FF', the printer sets the Data Check error indicator when the copy modification module is loaded. This error indicator can be blocked.

IEBIMAGE EXAMPLES

The following examples illustrate some of the uses of IEBIMAGE. Figure 76 can be used as a quick-reference guide to the examples that follow.

In most cases, examples for the 3800 Model 3 can be changed to 3800 Model 1 examples by deleting the OPTION DEVICE=3800M3 statement and specifying the OVERRUN parameter equal to a number other than 10. See the parameter charts for restrictions on the LPI parameter and on data statements.

Module Created	Printer	Comments	Example
FCB	3800 Model 1	11-inch form	1
FCB	3800 Model 1	5-1/2 inch form, replaces existing SYS1.IMAGELIB member. Multiple channel codes specified.	2
FCB	3800 Model 1	3-1/2 inch form, replaces existing SYS1.IMAGELIB member. Varied vertical spacing.	3
FCB	3800 Model 1	7-inch form, varied vertical spacing.	4
FCB	3800 Model 1	12-inch ISO form. Replaces IBM-supplied module.	5
FCB	3800 Model 3	7-1/2 inch ISO form. Varied vertical spacing.	6
FCB	4248	11-inch form, based on existing module. New print speed and copy position specified.	6A
COPYMOD	3800 Model 1	4 modification segments.	7
COPYMOD	3800 Model 3	Existing module used as basis for new module. OVERRUN specified.	8
TABLE	3800 Model 3	IBM-supplied module modified to include another character.	9
TABLE	3800 Model 3	Existing module used as basis for new module. Pitch changed.	10
TABLE	3800 Model 1	Existing module used as basis for new module. Includes user-designed characters of GRAPHIC module.	11
TABLE	3800 Model 3	Existing module used as basis for new module. New module deletes all GRAPHIC references and resets translation table entries.	12
GRAPHIC	3800 Model 1	Entire IBM-supplied module printed.	13
GRAPHIC	3800 Model 3	Segments copied from IBM-supplied module.	14
GRAPHIC	3800 Model 3	New module contains a user-designed character. Existing character arrangement (TABLE) modified to include new character.	15
GRAPHIC	3800 Model 1	Segments copied from existing module. User-designed character created.	16
GRAPHIC	3800 Model 3	New GRAPHIC module contains a user-designed character. Existing character arrangement (TABLE) modified to include new character. COPYMOD created to print new character. Result tested.	17

Figure 76 (Part 1 of 2). IEBIMAGE Example Directory

Module Created	Printer	Comments	Example
CHARSET	3800 Model 1	Entire library character set with scan patterns printed.	18
CHARSET	3800 Model 3	Segments copied from IBM-supplied GRAPHIC module.	19
CHARSET	3800 Model 3	New module contains a user-designed character. Existing character arrangement (TABLE) modified to include new character.	20
CHARSET	3800 Model 1	Segments copied from existing module. User-designed character created.	21

Figure 76 (Part 2 of 2). IEBIMAGE Example Directory

EXAMPLE 1: BUILDING A NEW 3800 FORMS CONTROL BUFFER MODULE

3800 Model 1

In this example, the vertical spacing and channel codes for an 11-inch form are specified, and the module is added to the SYS1.IMAGELIB data set as a new member.

```
//FCBMOD1   JOB   ...
//           EXEC  PGM=IEBIMAGE
//SYSUT1     DD    DSNAME=SYS1.IMAGELIB,DISP=OLD
//SYSPRINT   DD    SYSOUT=A
//SYSIN      DD    *
     FCB  CH1=1,CH12=80,LPI=8
     NAME IJ
/*
```

The control statements are discussed below.

- The SYSUT1 DD statement includes DISP=OLD to ensure that no other job can modify the data set while this job is executing.
- CH1=1 specifies channel 1 code for line 1, allowing for positioning at line 1.
- CH12=80 specifies channel 12 code for line 80, allowing for positioning at line 80 and a unit exception indication at line 80 (the last printable line on the page.)
- LPI=8 specifies that the entire form is to be at a vertical spacing of 8 lines per inch. Because the SIZE parameter is omitted, the form length defaults to 11 inches. Because there are 10 inches of printable space in an 11-inch form, 80 lines are printed at 8 lines per inch.
- The name of the new FCB module is IJ, and it is stored as a member of the SYS1.IMAGELIB data set.

EXAMPLE 2: REPLACING A 3800 FORMS CONTROL BUFFER MODULE

3800 Model 1

In this example, the size and channel codes for a 5-1/2 inch form are specified, and the module is added to the SYS1.IMAGELIB data set as a replacement for an existing member. The new module is added to the end of the data set; the name in the data set's directory is updated so that it points to the new module; the old module can no longer be accessed through the data set's directory.

```
//FCBMOD2  JOB  ...
//         EXEC PGM=IEBIMAGE
//SYSUT1   DD   DSNAME=SYS1.IMAGELIB,DISP=OLD
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
    FCB  CH1=(1,7,13,20),CH12=26,SIZE=55
    NAME S55(R)
/*
```

The control statements are discussed below.

- The SYSUT1 DD statement includes DISP=OLD to ensure that no other job can modify the data set while this job is executing.
- CH1=(1,7,13,20) specifies channel 1 code for printable line 1, line 7, line 13, and line 20.
- CH12=26 specifies channel 12 code for printable line 26.
- SIZE=55 specifies the length of the form as 55 tenths of an inch, or 5-1/2 inches.
- Because the LPI parameter is omitted, the vertical spacing defaults to 6 lines per inch. Because there are 4-1/2 inches of printable lines in a 5-1/2 inch form, there are 27 print lines on this form.
- The name of the FCB module is S55, and it replaces an existing FCB module of the same name. The new FCB module is stored as a member of the SYS1.IMAGELIB data set.

EXAMPLE 3: REPLACING A 3800 FORMS CONTROL BUFFER MODULE

3800 Model 1

In this example, the vertical spacing, channel codes, and size for a form are specified, and the module is added to the SYS1.IMAGELIB data set as a replacement for an existing member. The new module is added to the end of the data set; the name in the data set's directory is updated so that it points to the new module; the old module can no longer be accessed through the data set's directory.

```
                                                                      72
//FCBMOD3  JOB  ...
//         EXEC PGM=IEBIMAGE
//SYSUT1   DD   DSNAME=SYS1.IMAGELIB,DISP=OLD
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
    FCB  CH1=1,CH2=4,CH5=11,SIZE=35,                                  X
               LPI=((6,2),(8,3),(6,4),(8,9))
    NAME HL(R)
/*
```

The control statements are discussed below.

- The SYSUT1 DD statement includes DISP=OLD to ensure that no other job can modify the data set while this job is executing.
- CH1=1 specifies channel 1 code for printable line 1.
- CH2=4 specifies channel 2 code for line 4.
- CH5=11 specifies channel 5 code for line 11.
- LPI=((6,2),(8,3),(6,4),(8,9)) specifies vertical spacing for the first 18 printable lines in the form:

  (6,2) specifies lines 1 through 2 are at a vertical spacing of 6 lines per inch, and take up 2/6 inch.

  (8,3) specifies lines 3 through 5 are at a vertical spacing of 8 lines per inch, and take up 3/8 inch.

  (6,4) specifies lines 6 through 9 are at a vertical spacing of 6 lines per inch, and take up 4/6 inch.

  (8,9) specifies lines 10 through 18 are at a vertical spacing of 8 lines per inch, and take up 1-1/8 inch.
- SIZE=35 specifies the length of the form as 35 tenths of an inch, or 3-1/2 inches. Because there are 2-1/2 inches of printable space on a 3-1/2 inch form, and since the LPI parameter specifies vertical spacing for 2-1/2 inches of lines, the vertical spacing of all lines in the form is accounted for.
- The name of the FCB module is HL, and it replaces an existing module of the same name. The new FCB module is stored as a member of the SYS1.IMAGELIB data set.

EXAMPLE 4: BUILDING A NEW 3800 FORMS CONTROL BUFFER MODULE

3800 Model 1

In this example, the vertical spacing, channel codes, and length of a form are specified, and the module is added to the SYS1.IMAGELIB data set as a new member.

```
                                                                      72
//FCBMOD4  JOB  ...
//        EXEC PGM=IEBIMAGE
//SYSUT1   DD   DSNAME=SYS1.IMAGELIB,DISP=OLD
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *                                                     X
    FCB  CH1=1,CH6=33,SIZE=70,
               LPI=((8,32),(12,2))
    NAME TGT
/*
```

The control statements are discussed below.

- The SYSUT1 DD statement includes DISP=OLD to ensure that no other job can modify the data set while this job is executing.
- CH1=1 specifies channel 1 code for printable line 1.
- CH6=33 specifies channel 6 code for line 33.
- LPI=((8,32),(12,2)) specifies that the first 32 printable lines of the form are to be at a vertical spacing of 8 lines per inch, and the next 2 printable lines are to be at a vertical spacing of 12 lines per inch.
- SIZE=70 specifies that the length of the form is 70 tenths of an inch, or 7 inches. Because there are 6 inches of printable lines in a 7-inch form and the LPI parameter specifies 32 lines at 8 lines per inch, or 4 inches, and 2 lines at 12 lines per inch, or 1/6 inch, the vertical spacing for the remaining 1-5/6 inches defaults to 6 lines per inch.

  Therefore, the form consists of lines 1 through 32 at 8 lines per inch, lines 33 through 34 at 12 lines per inch, and lines 35 through 45 at 6 lines per inch, with channel codes at line 1 and line 33.
- The name of the new FCB module is TGT; it is stored as a member of the SYS1.IMAGELIB data set.

EXAMPLE 5: REPLACING THE 3800 FORMS CONTROL BUFFER MODULE STD3

3800 Model 1

In this example, an FCB module is defined that uses ISO paper sizes, replacing the IBM-supplied module named STD3. This must be done before the dump-formatting routines that print high-density dumps can print them at 8 lines per inch on that printer.

```
//FCBMOD5  JOB  ...
//        EXEC PGM=IEBIMAGE
//SYSUT1   DD   DSNAME=SYS1.IMAGELIB,DISP=OLD
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
    FCB  CH1=1,CH12=88,LPI=(8,88),SIZE=120
    NAME STD3(R)
/*
```

The control statements are discussed below.

- The SYSUT1 DD statement includes DISP=OLD to ensure that no other job can modify the data set while this job is executing.
- CH1=1 specifies channel 1 code for printable line 1; CH12=88 specifies channel 12 code for line 88.
- LPI=(8,88) specifies that all 88 printable lines of the form are to be at a vertical spacing of 8 lines per inch.
- SIZE=120 specifies that the length of the form is 120 tenths of an inch, or 12 inches, which is the longest ISO paper size.
- The name of the new FCB module is STD3, and it is to replace the existing module of that same name on SYS1.IMAGELIB.

EXAMPLE 6: BUILDING A NEW 3800 FORMS CONTROL BUFFER MODULE FOR ADDITIONAL ISO PAPER SIZES

3800 Model 3

In this example, an FCB module is defined that uses ISO paper sizes and has the ISO Paper Sizes Additional Feature installed.

```
                                                                      72
//FCBMOD6   JOB   ...
//           EXEC  PGM=IEBIMAGE
//SYSUT1     DD    DSNAME=SYS1.IMAGELIB,DISP=OLD
//SYSPRINT   DD    SYSOUT=A
//SYSIN      DD    *
    FCB  CH1=1,CH12=74,SIZE=75,                                        X
         LPI=((10,35),(12,4),(10,35),(6,1))
    NAME ARU
/*
```

The control statements are discussed below.

- The SYSUT1 DD statement includes DISP=OLD to ensure that no other job can modify the data set while this job is executing.
- CH1=1 specifies channel 1 code for line 1, allowing for positioning at line 1.
- CH12=74 specifies channel 12 code for line 74, allowing for positioning at line 74 and a unit exception indication at line 74 (the last printable line on the page.)
- LPI=((10,35),(12,4),(10,35),(6,1)) specifies vertical spacing for the entire printable area on the form. The last printable line on the form must have vertical spacing of 6 lines per inch.
- SIZE=75 specifies the length of the form as 75 tenths of an inch, or 7-1/2 inches, although the printable area is 7-1/3 inches.
- The name of the new FCB module is ARU, and it is stored as a member of the SYS1.IMAGELIB data set.

EXAMPLE 6A: BUILDING A 4248 FORMS CONTROL BUFFER MODULE

In this example, a new 4248 default FCB module is built using an existing FCB module as a model. The new module, NEW1, is added to SYS1.IMAGELIB as a new member. The existing module, OLD1, remains unchanged. OLD1 may be a 4248 FCB called FCB40LD1, or it may be a 3211 FCB called FCB20LD1. (If both modules existed, FCB40LD1 would be used.)

```
//FCBMOD7   JOB    ...
//           EXEC   PGM=IEBIMAGE
//SYSUT1     DD     DSNAME=SYS1.IMAGELIB,DISP=OLD
//SYSPRINT   DD     SYSOUT=A
//SYSIN      DD     *
     OPTION   DEVICE=4248
     INCLUDE  OLD1
     FCB      COPYP=67,PSPEED=M,DEFAULT=YES
     NAME     NEW1
/*
```

The control statements are discussed below.

- The SYSUT1 DD statement includes DISP=OLD to ensure that no other job can modify the data set while this job is executing.
- DEVICE=4248 on the OPTION statement specifies that this module is to be created for the 4248 printer.
- The INCLUDE statement specifies that a copy of the existing module OLD1 is to be used as a basis for the new module, NEW1.
- COPYP=67 indicates that the horizontal copy feature should be activated, and that horizontal copies should begin printing in the 67th print position from the left margin. This setting overrides any COPYP value previously set in module OLD1; it applies to module NEW1, but does not change the value set in OLD1.

  Note that the value 67 divides a 132-hammer printer into two equal copy areas for two equally-sized horizontal copies. With COPYP=67, a maximum of 66 bytes can be sent to the printer.
- PSPEED=M indicates that the printer speed should be set to medium (3000 LPM). This setting overrides any PSPEED value previously set in module OLD1; it applies to module NEW1, but does not change the value set in OLD1.
- DEFAULT=YES indicates that this module, NEW1, should become a default FCB module for this installation.
- Because these parameters are not specified, the values of LINES, SIZE, LPI, and CHx default to the values which already exist in module OLD1.
- The NAME statement indicates that this module should be called NEW1.

EXAMPLE 7: BUILDING A NEW COPY MODIFICATION MODULE

3800 Model 1

In this example, a copy modification module that contains four modification segments is built. The module is added to the SYS1.IMAGELIB data set as a new member.

```
                                                                      72
//COPMOD1  JOB  ...
//        EXEC PGM=IEBIMAGE
//SYSUT1   DD   DSNAME=SYS1.IMAGELIB,DISP=OLD
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
COPY1  COPYMOD COPIES=(1,1),                                          X
               LINES=(1,1),POS=50,                                    X
               TEXT=(C,'CONTROLLER''S COPY')
COPY2A COPYMOD COPIES=(2,1),                                          X
               LINES=(1,1),POS=50,                                    X
               TEXT=(C,'SHIPPING MANAGER''S COPY')
COPY2B COPYMOD COPIES=(2,1),                                          X
               LINES=(34,3),POS=75,                                   X
               TEXT=(10C,' ')
COPYALL COPYMOD COPIES=(1,4),                                         X
               LINES=(58,1),POS=35,                                   X
               TEXT=((C,'***'),(C,'CONFIDENTIAL'),                    X
               (3X,'5C'))
        NAME   RT01
/*
```

The control statements are discussed below.

- The SYSUT1 DD statement includes DISP=OLD to ensure that no other job can modify the data set while this job is executing.

- The COPY1 COPYMOD statement specifies text that applies to each page of the first copy of the output data set:

  LINES=(1,1) and POS=50 specify that the text is to be on the first printable line of each page, starting at the 50th print position from the left.

  The TEXT parameter identifies each page of the copy as being the "Controller's Copy."

- The COPY2A COPYMOD statement specifies text that applies to each page of the second copy of the output data set. The text is to be on the first line of each page, at the 50th print position from the left, with each page of the copy being the "Shipping Manager's Copy."

- The COPY2B COPYMOD statement specifies that part of the second copy's output data set text is to be blanked out, so that the first, third, and subsequent copies contain information that is not printed on the second copy. The blank area is to be on lines 34, 35, and 36, beginning at the 75th print position from the left. The text on lines 34, 35, and 36, between print positions 75 and 84, is to be blank (that is, the character specified between the TEXT parameter's single quotation marks is a blank).

- The COPYALL COPYMOD statement specifies text that applies to the first four copies of the output data set. This example assumes that no more than four copies are printed each time the job that produces the output data set is executed. The text is to be on the 58th line on each page, at the 35th

print position from the left. The legend "***CONFIDENTIAL***" is to be on each page of the copy. Note that the text can be coded in both character and hexadecimal format.

- The name of the copy modification module is RTO1, and it is stored as a member of the SYS1.IMAGELIB data set.

EXAMPLE 8: BUILDING A NEW COPY MODIFICATION MODULE FROM AN EXISTING COPY

3800 Model 3

In this example, a copy of an existing copy modification module, RTO1, is used as the basis for a new copy modification module. The new module is added to the SYS1.IMAGELIB data set as a new member. The existing module, RTO1, remains unchanged and available for use.

```
                                                                      72
//COPMOD2  JOB  ...
//         EXEC PGM=IEBIMAGE
//SYSUT1   DD   DSNAME=SYS1.IMAGELIB,DISP=OLD
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
    INCLUDE  RTO1,DELSEG=1
    OPTION   OVERRUN=8,DEVICE=3800M3
    COPYMOD  COPIES=(2,3),                                             X
             LINES=(52,6),POS=100,                                     X
             TEXT=(X,'40404040404040405C5C')
    NAME     AP
/*
```

The control statements are discussed below.

- The SYSUT1 DD statement includes DISP=OLD to ensure that no other job can modify the data set while this job is executing.

- The INCLUDE statement specifies that a copy of the copy modification module named RTO1 is used as a basis for the new module, and that the first modification segment of RTO1 is to be deleted from the copy.

- OVERRUN=8 in the OPTION statement specifies that the IEBIMAGE program is to print a warning message if the copy modification could cause a line overrun condition when printing at 6 and 8 lines per inch. The program is also to suppress any warning messages that apply to printing at 10 and 12 lines per inch. DEVICE=3800M3 in the OPTION statement specifies 3800 Model 3 compatibility mode processing.

- The COPYMOD statement specifies text that applies to each page of the second, third, and fourth copies of the output data set:

  LINES=(52,6) and POS=100 specify that the text is to be on the 52nd line and repeated for the 53rd through 57th lines of each page, starting at the 100th print position from the left.

  The TEXT statement specifies the text in hexadecimal form: eight blanks followed by two asterisks (in this example, the assumption is made that X'40' prints as a blank and that X'5C' prints as an asterisk; in actual practice, the character arrangement table used with the copy modification

module might translate X'40' and X'5C' to other printable characters).

- The name of the new copy modification module is AP, and it is stored as a member of the SYS1.IMAGELIB data set.

EXAMPLE 9: ADDING A NEW CHARACTER TO A CHARACTER ARRANGEMENT TABLE MODULE

3800 Model 3

In this example, an IBM-supplied character arrangement table module is modified to include another character, and then added to the SYS1.IMAGELIB data set as a replacement for the IBM-supplied module.

```
//CHARMOD1 JOB  ...
//         EXEC PGM=IEBIMAGE
//SYSUT1   DD   DSNAME=SYS1.IMAGELIB,DISP=OLD
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
     INCLUDE  GF10
     OPTION   DEVICE=3800M3
     TABLE    LOC=((2A,2A),(6A,2A),(AA,2A),(EA,2A))
     NAME     GF10(R)
/*
```

The control statements are discussed below.

- The SYSUT1 DD statement includes DISP=OLD to ensure that no other job can modify the data set while this job is executing.
- The INCLUDE statement specifies that a copy of the character arrangement table named GF10 is to be used as a basis for the new module.
- The OPTION statement with the DEVICE parameter specifies 3800 Model 3 compatibility mode processing.
- The TABLE statement specifies updated information for four translate table entries: X'2A', X'6A', X'AA', and X'EA'. (These four locations are unused in the IBM-supplied GF10 table.) Each of the four translate table entries is to point to the '2A' (43rd character) position in the first WCGM, which contains the scan pattern for a lozenge.
- The name of the character arrangement table is GF10, and it is stored as a new module in the SYS1.IMAGELIB data set. The data set's directory is updated so that the name GF10 points to the new module; the old GF10 module can no longer be accessed through the data set's directory.

EXAMPLE 10: BUILDING A NEW CHARACTER ARRANGEMENT TABLE MODULE FROM AN EXISTING COPY

3800 Model 3

In this example, an existing character arrangement table module is copied and used as a basis for a new module. The new character arrangement table is identical to the old one, except that it uses the Gothic 15-pitch character set instead of Gothic 10-pitch.

```
//CHARMOD2 JOB  ...
//        EXEC PGM=IEBIMAGE
//SYSUT1   DD   DSNAME=SYS1.IMAGELIB,DISP=OLD
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
    INCLUDE  A11
    OPTION   DEVICE=3800M3
    TABLE    CGMID=87
    NAME     A115
/*
```

The control statements are discussed below.

- The SYSUT1 DD statement includes DISP=OLD to ensure that no other job can modify the data set while this job is executing.
- The INCLUDE statement specifies that a copy of the character arrangement table named A11 is to be used as a basis for the new module. The A11 character arrangement table translates 8-bit data codes to printable characters in the Gothic 10-pitch character set.
- The OPTION statement with the DEVICE parameter specifies 3800 Model 3 compatibility mode processing.
- The TABLE statement specifies a new character set identifier, X'87', which is the identifier for the Gothic 15-pitch character set. No other changes are made to the character arrangement table. The new table calls for characters in the Gothic 15-pitch character set.
- The name of the new character arrangement table is A115, and it is stored as a member of the SYS1.IMAGELIB data set.

EXAMPLE 11: BUILDING GRAPHIC CHARACTERS IN A CHARACTER ARRANGEMENT TABLE MODULE

3800 Model 1

In this example, an existing character arrangement table module is copied and used as the basis for a new module that will include user-designed characters of a graphic character modification module. The new module is then added to the SYS1.IMAGELIB data set.

```
                                                                      72
//CHARMOD3 JOB  ...
//        EXEC PGM=IEBIMAGE
//SYSUT1   DD   DSNAME=SYS1.IMAGELIB,DISP=OLD
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
    INCLUDE ONB
    TABLE   GCMLIST=ONB1,                                             X
            LOC=((6F,2F,1),(7C,3C,1),(6A,2A,0))
    NAME    ONBZ
/*
```

The control statements are discussed below.

- The SYSUT1 DD statement includes DISP=OLD to ensure that no other job can modify the data set while this job is executing.
- The INCLUDE statement specifies that a copy of the character arrangement table named ONB is to be used as a basis for the new module. ONB refers to two WCGMs.
- The TABLE statement identifies a graphic character modification module and stipulates the translate table entries for each of its segments:

  GCMLIST=ONB1 identifies the graphic character modification module named ONB1. The LOC parameter specifies the translate table entry location, character position, and WCGM number for each segment of the module:

  The first segment corresponds to the 8-bit data code X'6F'. The segment's scan pattern is to be loaded at character position X'2F' (that is, the 48th character position) in the second WCGM.

  The second segment corresponds to the 8-bit data code X'7C'. The segment's scan pattern is to be loaded at character position X'3C' (that is, the 61st character position) in the second WCGM.

  The third segment corresponds to the 8-bit data code X'6A'. The segment's scan pattern is to be loaded at character position X'2A' (that is, the 43rd character position) in the first WCGM.
- The name of the new character arrangement table is ONBZ, and it is stored as a new module in the SYS1.IMAGELIB data set.

EXAMPLE 12: DELETING GRAPHIC REFERENCES FROM A CHARACTER ARRANGEMENT TABLE MODULE

3800 Model 3

In this example, an existing character arrangement table module is copied and used as a basis for a new one. The new character arrangement table deletes references to all graphic character modification modules and resets the translate table entries that were used to point to character positions for the segments of a graphic character modification module.

```
//CHARMOD4 JOB  ...
//         EXEC PGM=IEBIMAGE
//SYSUT1   DD   DSNAME=SYS1.IMAGELIB,DISP=OLD
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
    INCLUDE  ZYL
    OPTION   DEVICE=3800M3
    TABLE    GCMLIST=DELETE,LOC=((6A),(6B))
    NAME     ZYLA
/*
```

The control statements are discussed below.

- The SYSUT1 DD statement includes DISP=OLD to ensure that no other job can modify the data set while this job is executing.
- The INCLUDE statement specifies that a copy of the character arrangement table named ZYL is to be used as a basis for the new module.

- The OPTION statement with the DEVICE parameter specifies 3800 Model 3 compatibility mode processing.
- The TABLE statement deletes references to graphic character modification modules and resets two translate table entries:

  GCMLIST=DELETE specifies that all names of graphic character modification modules included with the module when the ZYL character arrangement table was copied are to be reset to blanks (X'40').

  The LOC parameter identifies two locations in the translate table, X'6A' and X'6B', that are to be set to X'FF' (the default value when no character position or WCGM values are specified).
- The name of the new character arrangement table is ZYLA, and it is stored as a member of the SYS1.IMAGELIB data set.

EXAMPLE 13: LISTING THE WORLD TRADE NATIONAL USE GRAPHICS GRAPHIC CHARACTER MODIFICATION MODULE

3800 Model 1

In this example, each segment of the IBM-supplied graphic character modification module containing the World Trade National Use Graphics is printed. Each segment is unique, although the scan patterns for some segments are identical to other segment's scan patterns with only the 8-bit data code being different.

```
//GRAFMOD1 JOB  ...
//        EXEC PGM=IEBIMAGE
//SYSUT1   DD   DSNAME=SYS1.IMAGELIB,DISP=SHR
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
    GRAPHIC
    NAME  *
/*
```

The control statements are discussed below.

- DISP=SHR is coded because the library is not being updated.
- The World Trade National Use Graphics graphic character modification module is identified with the pseudonym of "*". The scan pattern of each of the characters in the module is printed.

EXAMPLE 14: BUILDING A GRAPHIC CHARACTER MODIFICATION MODULE FROM THE WORLD TRADE GRAFMOD

3800 Model 3

In this example, a graphic character modification module is built. Its characters are segments copied from the World Trade National Use Graphics graphic character modification module. (See the IBM 3800 Printing Subsystem Programmer's Guide for the EBCDIC assignments for the characters.) The new module is stored in the SYS1.IMAGELIB system data set.

```
                                                                      72
//GRAFMOD2 JOB  ...
//         EXEC PGM=IEBIMAGE
//SYSUT1   DD   DSNAME=SYS1.IMAGELIB,DISP=OLD
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
    OPTION DEVICE=3800M3
    GRAPHIC  REF=((24),(25),(26),(27),(28),                          X
               (31),(33),(35),(38),(40))
    NAME     CSTW
/*
```

The control statements are discussed below.

- The SYSUT1 DD statement includes DISP=OLD to ensure that no other job can modify the data set while this job is executing.
- DEVICE=3800M3 in the OPTION statement specifies 3800 Model 3 compatibility mode module format.
- By not specifying the GCM keyword, the GRAPHIC statement identifies the World Trade National Use Graphics graphic character modification module. Ten of its segments are to be copied and used with the new module.
- The name of the graphic character modification module is CSTW, and it is stored as a new module in the SYS1.IMAGELIB data set.

EXAMPLE 15: BUILDING A NEW GRAPHIC CHARACTER MODIFICATION MODULE AND MODIFYING A CHARACTER ARRANGEMENT TABLE TO USE IT

3800 Model 3

In this example, a graphic character modification module is built. The module contains one user-designed character, a reverse 'E', whose 8-bit data code is designated as X'E0' and whose pitch is 10. An existing character arrangement table is then modified to include the reverse E.

```
//GRAFMOD3 JOB  ...
//         EXEC PGM=IEBIMAGE
//SYSUT1   DD   DSNAME=SYS1.IMAGELIB,DISP=OLD
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
         OPTION  DEVICE=3800M3
         GRAPHIC  ASSIGN=(E0,10)
    XXXXXXXXXXXXXXX      SEQ=10
    XXXXXXXXXXXXXXX      SEQ=11
    XXXXXXXXXXXXXXX      SEQ=12
                XXXX      SEQ=13
                XXXX      SEQ=14
                XXXX      SEQ=15
                XXXX      SEQ=16
                XXXX      SEQ=17
                XXXX      SEQ=18
                XXXX      SEQ=19
      XXXXXXXXXXXXX      SEQ=20
      XXXXXXXXXXXXX      SEQ=21
      XXXXXXXXXXXXX      SEQ=22
                XXXX      SEQ=23
                XXXX      SEQ=24
                XXXX      SEQ=25
                XXXX      SEQ=26
                XXXX      SEQ=27
                XXXX      SEQ=28
                XXXX      SEQ=29
    XXXXXXXXXXXXXXX      SEQ=30
    XXXXXXXXXXXXXXX      SEQ=31
    XXXXXXXXXXXXXXX      SEQ=32
    NAME      BODE
    INCLUDE   GS10
    OPTION    DEVICE=3800M3
    TABLE     CGMID=(83,FF),
              GCMLIST=BODE,
              LOC=(E0,03,1)
    NAME      RE10
/*
```

The control statements are discussed below.

- The SYSUT1 DD statement includes DISP=OLD to ensure that no other job can modify the data set while this job is executing.
- DEVICE=3800M3 in the OPTION statement preceding the GRAPHIC statement specifies 3800 Model 3 compatibility mode processing.
- The GRAPHIC statement's ASSIGN parameter establishes the 8-bit data code, X'E0', and the width, 10-pitch, for the user-designed character. The data statements that follow the GRAPHIC statement describe the character's scan pattern.
- The name of the graphic character modification module is BODE, and it is stored as a new module in the SYS1.IMAGELIB data set.
- The INCLUDE statement specifies that a copy of the GS10 character arrangement table is to be used as the basis for the new table.
- The TABLE statement specifies the addition of the reverse E to that copy of the GS10 table.

  CGMID=(83,FF) specifies the character set identifier X'83' for the Gothic-10 set (which is the set already used by the GS10 table) and specifies X'FF' as a character set

identifier to allow accessing of the second WCGM without loading it.

GCMLIST=BODE identifies the graphic character modification module containing the reverse E for inclusion in the table.

LOC=(E0,03,1) specifies that the reverse E, which has been assigned the 8-bit data code X'E0', is to be loaded into position X'03' in the second WCGM. Because this second WCGM is otherwise unused, any position in it could have been used for the reverse E.

- The new character arrangement table is named RE10 and stored as a new module in SYS1.IMAGELIB.

EXAMPLE 16: BUILDING A GRAPHIC CHARACTER MODIFICATION MODULE FROM MULTIPLE SOURCES

3800 Model 1

In this example, a graphic character modification module is created. Its contents come from three different sources: nine segments are copied from an existing module with the INCLUDE statement; the GRAPHIC statement is used to select another segment to be copied; the GRAPHIC statement is also used to establish characteristics for a user-designed character. The new graphic character modification module, when built, is added to the SYS1.IMAGELIB.

```
//GRAFMOD4 JOB  ...
//         EXEC PGM=IEBIMAGE
//SYSUT1   DD   DSNAME=SYS1.IMAGELIB,DISP=OLD
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
     INCLUDE CSTW,DELSEG=3
     GRAPHIC REF=(1,6A),GCM=BODE,ASSIGN=9A
      ********            SEQ=06
     **********           SEQ=07
    ****    ****          SEQ=08
    ***      ***          SEQ=09
    ***     ****          SEQ=10
    ***  ******           SEQ=11
    ***  ******           SEQ=12
    ***     ****          SEQ=13
    ***      ****         SEQ=14
    ***       ***         SEQ=15
    ***       ***         SEQ=16
    *** **** ****         SEQ=17
    ***  *******          SEQ=18
    ***   *****           SEQ=19
     NAME  JPCK
/*
```

The control statements are discussed below.

- The SYSUT1 DD statement includes DISP=OLD to ensure that no other job can modify the data set while this job is executing.

- The INCLUDE statement specifies that a copy of the graphic character modification module named CSTW is to be included with the new module. All segments of CSTW, except the third segment (as a result of DELSEG=3), are to be copied into the new module and become the module's first through ninth modification segments.

- The GRAPHIC statement specifies the module's tenth and eleventh segments:

  REF=(1,6A) and GCM=BODE specify that the tenth segment of the new module is to be obtained by copying the first segment from the graphic character modification module named BODE. In addition, the segment's 8-bit data code is to be changed so that its character is identified with the code X'6A'.

  ASSIGN=9A specifies that the new module's eleventh segment is a user-designed character whose 8-bit data code is X'9A' and whose width is 10-pitch (the default when no pitch value is specified). The GRAPHIC statement is followed by data statements that specify the character's scan pattern.

- The name of the graphic character modification module is JPCK, and it is stored as a new module in the SYS1.IMAGELIB data set.

EXAMPLE 17: DEFINING AND USING A CHARACTER IN A GRAPHIC CHARACTER MODIFICATION MODULE

3800 Model 3

In this example, a graphic character modification module containing a user-designed character is built. Next, a Format character arrangement table is modified to include that new character. Then, a copy modification module is created to print the new character enclosed in a box of Format characters. Finally, the result is tested to allow comparison of the output with the input.

```
//CHAR     JOB  ...
//BUILD    EXEC PGM=IEBIMAGE
//SYSUT1   DD   DSNAME=SYS1.IMAGELIB,DISP=OLD
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
         OPTION  DEVICE=3800M3
STEP1    GRAPHIC ASSIGN=5C
XXX                   XXX SEQ=01
XXX                   XXX SEQ=02
XXX                   XXX SEQ=03
XXX                   XXX SEQ=04
XXXXXXXXXXXXXXXXXXXXX SEQ=05
XXXXXXXXXXXXXXXXXXXXX SEQ=06
XXXXXXXXXXXXXXXXXXXXX SEQ=07
XXX                   XXX SEQ=08
XXX                   XXX SEQ=09
XXX                   XXX SEQ=10
XXX                   XXX SEQ=11
                          SEQ=12
                          SEQ=13
                          SEQ=14
XXXXXXXXXXXXXXXXXXXXX SEQ=15
XXXXXXXXXXXXXXXXXXXXX SEQ=16
XXXXXXXXXXXXXXXXXXXXX SEQ=17
XXX        XXX        XXX SEQ=18
XXX        XXX        XXX SEQ=19
XXX        XXX        XXX SEQ=20
XXX        XXX        XXX SEQ=21
XXXX      XXXXX      XXXX SEQ=22
 XXXX   XXXXXXX   XXXX  SEQ=23
  XXXXXXXXXXXXXXXXXX   SEQ=24
    XXXXX    XXXXXX     SEQ=25
                          SEQ=26
                          SEQ=27
                          SEQ=28
XXXXXXXXXXXXXXXXXXXXX SEQ=29
XXXXXXXXXXXXXXXXXXXXX SEQ=30
XXXXXXXXXXXXXXXXXXXXX SEQ=31
                  XXXXXX SEQ=32
     XXXXXXXXXXXXXXXXX SEQ=33
XXXXXXXXXXXXXXX       SEQ=34
XXXXXXXXXXXXXXX       SEQ=35
     XXXXXXXXXXXXXXXXX SEQ=36
                  XXXXXX SEQ=37
XXXXXXXXXXXXXXXXXXXXX SEQ=38
XXXXXXXXXXXXXXXXXXXXX SEQ=39
XXXXXXXXXXXXXXXXXXXXX SEQ=40

         NAME AIBM
```

```
                                                                      72
STEP2    OPTION  DEVICE=3800M3
         INCLUDE FM10
         TABLE   GCMLIST=AIBM,LOC=(5C,2C)
         NAME    BIBM
STEP3    OPTION  DEVICE=3800M3
         COPYMOD COPIES=1,LINES=58,POS=5,                             X
                 TEXT=(C,'W6X')
         COPYMOD COPIES=1,LINES=59,POS=5,                             X
                 TEXT=(C,'7*7')
         COPYMOD COPIES=1,LINES=60,POS=5,                             X
                 TEXT=(X,'E9F6E8')
         NAME    CIBM
/*
//TEST     EXEC PGM=IEBIMAGE
//SYSUT1   DD   DSNAME=SYS1.IMAGELIB,DISP=OLD
//SYSPRINT DD   SYSOUT=A,CHARS=(GF10,BIBM),
//             MODIFY=(CIBM,1)
//SYSIN    DD   *
        OPTION   DEVICE=3800M3
        GRAPHIC
        NAME     AIBM
/*
```

The control statements are discussed below.

- The SYSUT1 DD statement includes DISP=OLD to ensure that no other job can modify the data set while this job is executing.
- The GRAPHIC statement's ASSIGN parameter specifies that the 8-bit data code for the user-designed character is X'5C' and the width is 10-pitch (the default when no pitch is specified). The GRAPHIC statement is followed by data statements that specify the character's scan pattern for vertical line spacing of 6 lines per inch.
- The name of the graphic character modification module is AIBM, and it is stored as a new module in SYS1.IMAGELIB.
- At STEP2, the INCLUDE statement specifies that a copy of the FM10 character arrangement table is to be used as a basis for the new module.
- The TABLE statement identifies the graphic character modification module named AIBM, created in the previous step. The TABLE statement's LOC parameter specifies the translate table entry location (the character's 8-bit data code) of X'5C' and the position (X'2C') where that character is to be loaded into the WCGM.
- The name of the new character arrangement table, which is added to SYS1.IMAGELIB, is BIBM.
- At STEP3, the three COPYMOD statements specify text that is to be placed on lines 58, 59, and 60 of the first copy of the output data set, starting at print position 5 on each line. When used with the BIBM character arrangement table, the characters W, 6, and X print as a top left corner, horizontal line segment, and top right corner, all in line weight 3. The characters 7, *, and 7 print as a weight-3 vertical line segment on both sides of the user-designed character built at STEP1 (the asterisk has the EBCDIC assignment 5C, which addresses that character). The hexadecimal E9, F6, and E8 complete the line-weight-3 Format box around the character.
- The name of the copy modification module is CIBM, and it is stored as a new module on SYS1.IMAGELIB.

- At TEST, the EXEC statement calls for another execution of the IEBIMAGE program to test the modules just created. On the SYSPRINT DD statement the BIBM character arrangement table is the second of two specified, and the CIBM copy modification module is specified with a table reference character of 1, to use that BIBM table.
- The GRAPHIC statement with no operand specified calls for printing of the module, AIBM, specified with the NAME statement that follows it. Each page of the output listing for this IEBIMAGE run has the following modification printed in the lower left corner:

- The OPTION statement with the DEVICE parameter at STEP1, STEP2, and STEP3 specifies 3800 Model 3 compatibility mode module format and processing considerations.

EXAMPLE 18: LISTING A LIBRARY CHARACTER SET MODULE

3800 Model 1

In this example, each segment of a library character set is printed. The scan pattern of each of the characters in the module is printed.

```
//LIBMOD1  JOB  ...
//         EXEC PGM=IEBIMAGE
//SYSUT1   DD   DSNAME=SYS1.IMAGELIB,DISP=SHR
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
    CHARSET
    NAME    83
/*
```

The control statements are discussed below.

- NAME specifies the name of the library character set (83).

EXAMPLE 19: BUILDING A LIBRARY CHARACTER SET MODULE

3800 Model 3

In this example, a library character set module is built. Its characters are segments copied from the World Trade National Use Graphics graphic character modification module. (See the IBM 3800 Printing Subsystem Programmer's Guide for the listing of all the segments of that module. The EBCDIC assignments for the characters are replaced by WCGM-location codes.) The new module is stored in the SYS1.IMAGELIB system data set.

```
                                                                      72
//LIBMOD2  JOB  ...
//         EXEC PGM=IEBIMAGE
//SYSUT1   DD   DSNAME=SYS1.IMAGELIB,DISP=OLD
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
    OPTION   DEVICE=3800M3
    CHARSET  REF=((24,01),(25,02),(26,03),(27,04),(28,05), X
               (31,06),(33,07),(35,08),(38,09),(40,0A))
    NAME     73
/*
```

The control statements are discussed below.

- The SYSUT1 DD statement includes DISP=OLD to ensure that no other job can modify the data set while this job is executing.
- DEVICE=3800M3 in the OPTION statement specifies 3800 Model 3 compatibility mode module format.
- By not specifying the GCM keyword or a library character set ID, the CHARSET statement identifies the World Trade National Use Graphics graphic character modification module. Ten of its segments are to be copied and used with the new module. For example, the 24th segment is to be copied and assigned the WCGM location 01. See the REF parameter (24,01).
- The name of the library character set module is 73, and it is stored as a new module in the SYS1.IMAGELIB data set.

EXAMPLE 20: BUILDING A LIBRARY CHARACTER SET MODULE AND MODIFYING A CHARACTER ARRANGEMENT TABLE TO USE IT

3800 Model 3

In this example, a library character set module is built. The module contains one user-designed character, a reverse 'E', whose 6-bit WCGM-location code is designated as X'03', and whose pitch is 10. An existing character arrangement table is then modified to include the reverse E.

```
//LIBMOD3  JOB  ...
//        EXEC PGM=IEBIMAGE
//SYSUT1   DD   DSNAME=SYS1.IMAGELIB,DISP=OLD
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
    OPTION  DEVICE=3800M3
    CHARSET ASSIGN=(03,10)
    XXXXXXXXXXXXXX      SEQ=10
    XXXXXXXXXXXXXX      SEQ=11
    XXXXXXXXXXXXXX      SEQ=12
              XXXX      SEQ=13
              XXXX      SEQ=14
              XXXX      SEQ=15
              XXXX      SEQ=16
              XXXX      SEQ=17
              XXXX      SEQ=18
              XXXX      SEQ=19
      XXXXXXXXXXXXX     SEQ=20
      XXXXXXXXXXXXX     SEQ=21
      XXXXXXXXXXXXX     SEQ=22
              XXXX      SEQ=23
              XXXX      SEQ=24
              XXXX      SEQ=25
              XXXX      SEQ=26
              XXXX      SEQ=27
              XXXX      SEQ=28
              XXXX      SEQ=29
    XXXXXXXXXXXXXX      SEQ=30
    XXXXXXXXXXXXXX      SEQ=31
    XXXXXXXXXXXXXX      SEQ=32
    NAME     73
    INCLUDE  GS10
    OPTION   DEVICE=3800M3
    TABLE    CGMID=(83,73),LOC=(E0,03,1)
    NAME     RE10
/*
```

The control statements are discussed below.

- The SYSUT1 DD statement includes DISP=OLD to ensure that no other job can modify the data set while this job is executing.
- DEVICE=3800M3 in the OPTION statement specifies 3800 Model 3 compatibility mode module format and processing considerations.
- The CHARSET statement's ASSIGN parameter establishes the 6-bit WCGM-location code, X'03', and the width, 10-pitch, for the user-designed character. The data statements that follow the CHARSET statement describe the character's scan pattern.
- The name of the library character set module is 73, and it is stored as a new module in the SYS1.IMAGELIB data set.
- The INCLUDE statement specifies that a copy of the GS10 character arrangement table is to be used as the basis for the new table.
- The TABLE statement specifies the addition of the library character set containing the reverse E to that copy of the GS10 table.

  CGMID=(83,73) specifies the character set identifier X'83' for the Gothic-10 set (which is the set already used by the GS10 table) and specifies X'73' as a character set identifier to allow loading of the second WCGM with the library character set 73.

LOC=(E0,03,1) specifies that the reverse E, which has been assigned the WCGM location 03 in the second WCGM, is to be referenced by the EBCDIC code X'E0'.

The new character arrangement table is named RE10 and stored as a new module in SYS1.IMAGELIB.

EXAMPLE 21: BUILDING A LIBRARY CHARACTER SET MODULE FROM MULTIPLE SOURCES

3800 Model 1

In this example, a library character set module is created. Its contents come from three different sources: 62 segments are copied from an existing module with the INCLUDE statement; the CHARSET statement is used to select another segment to be copied; a second CHARSET statement is used to establish characteristics for a user-designed character. The new library character set module, when built, is added to the SYS1.IMAGELIB.

```
//LIBMOD4  JOB  ...
//        EXEC PGM=IEBIMAGE
//SYSUT1   DD   DSNAME=SYS1.IMAGELIB,DISP=OLD
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
    INCLUDE  33,DELSEG=(3,4)
    CHARSET  REF=(1,02),GCM=BODE,ASSIGN=03
     ********            SEQ=06
    **********           SEQ=07
   ****    ****          SEQ=08
   ***      ***          SEQ=09
   ***     ****          SEQ=10
   ***  ******           SEQ=11
   ***  ******           SEQ=12
   ***     ****          SEQ=13
   ***      ****         SEQ=14
   ***       ***         SEQ=15
   ***       ***         SEQ=16
   *** **** ****         SEQ=17
   ***  *******          SEQ=18
   ***   *****           SEQ=19
    NAME  53
/*
```

The control statements are discussed below.

- The SYSUT1 DD statement includes DISP=OLD to ensure that no other job can modify the data set while this job is executing.
- The INCLUDE statement specifies that a copy of the library character set module named 33 is to be included with the new module. All segments of 33, except the third and fourth segments (as a result of DELSEG=3,4), are to be copied into the new module and become the basis for the new module.
- The CHARSET statement specifies the module's third and fourth segments:

  REF=(1,02) and GCM=BODE specify that the third segment of the new module is to be obtained by copying the first segment from the graphic character modification module named BODE. The segment's 6-bit WCGM-location code is to be set so that its character is identified with the code X'02'.

ASSIGN=03 specifies that the new module's fourth segment is a user-designed character whose 6-bit WCGM-location code is X'03' and whose width is 10-pitch (the default when no pitch value is specified). The CHARSET statement is followed by data statements that specify the character's scan pattern.

- The name of the library character set module is 53, and it is stored as a new module in the SYS1.IMAGELIB data set.

IEBISAM can be used to:

- Copy an indexed sequential (ISAM) data set directly from one DASD volume to another.
- Create a backup (transportable) copy of an ISAM data set by copying (unloading) it into a sequential data set on a DASD or magnetic tape volume.
- Create an ISAM data set from an unloaded data set. The sequential (unloaded) data set is in a form that can be subsequently loaded, that is, it can be converted back into an ISAM data set.
- Print an ISAM data set.

COPYING AN ISAM DATA SET

IEBISAM can be used to copy an indexed sequential (ISAM) data set directly from one DASD volume to another. When the data set is copied, the records marked for deletion are only deleted if the DELETE parameter was specified in the OPTCD (optional control program service) field. Those records that are contained in the overflow area of the original data set are moved into the primary area of the copied data set. Control information characteristics such as BLKSIZE and OPTCD can be overridden by new specifications. Caution should be used, however, when overriding these characteristics (see "Overriding DCB Control Information" on page 232).

CREATING A SEQUENTIAL BACKUP COPY

An **unloaded** sequential data set can be created to serve as a backup or transportable copy of source data from an ISAM data set. Records marked for deletion within the ISAM data set are automatically deleted when the unloaded data set is created. When the data set is subsequently **loaded**—reconstructed into an ISAM data set—records that were contained in the overflow area assigned to the original data set are moved sequentially into the primary area.

An unloaded data set consists of 80-byte logical records. The data set contains:

- Fixed records from an ISAM data set
- Control information used in the subsequent loading of the data set

Control information consists of characteristics that were assigned to the ISAM data set. These characteristics are:

- Optional control program service (OPTCD)
- Record format (RECFM)
- Logical record length (LRECL)
- Block size (BLKSIZE)
- Relative key position (RKP)
- Number of tracks in master index (NTM)
- Key length (KEYLEN)
- Number of overflow tracks on each cylinder (CYLOFL)

OVERRIDING DCB CONTROL INFORMATION

When a load operation is specified, control information characteristics can be overridden by specifications in the DCB parameter of the SYSUT2 DD statement (refer to "Job Control Statements" on page 235 for a discussion of the SYSUT2 DD statement). Caution should be used, however, because checks are made to ensure that:

1. Record format is the same as that of the original indexed sequential data set (either fixed (F) or variable (V) length).

2. Logical record length is greater than or equal to that of the original ISAM data set when the RECFM is variable (V) or variable blocked (VB).

3. For fixed records, the block size is equal to or a multiple of the logical record length of the records in the original indexed sequential data set. For variable records, the block size is equal to or greater than the logical record length plus four.

4. Relative key position is equal to or less than the logical record length minus the key length. Following are relative key position considerations:

   - If the RECFM is V or VB, the relative key position should be at least 4.
   - If the DELETE parameter was specified in the OPTCD field and the RECFM is F or fixed blocked (FB), the relative key position should be at least 1.
   - If the DELETE parameter was specified in the OPTCD field and the RECFM is V or VB, the relative key position should be at least 5.

5. The key length is less than or equal to 255 bytes.

6. For a fixed unblocked data set with RKP=0, the LRECL value is the length of the data portion, not, as in all other cases, the data portion and key length. When changing an RKP=0 data set RECFM from fixed unblocked and to fixed blocked, the new LRECL must be equal to the old LRECL **plus** the old key length.

   If either RKP or KEYLEN is overridden, it might not be possible to reconstruct the data set.

   The number of 80-byte logical records in an unloaded data set can be approximated by the following formula:

$$x = \frac{n(y+2) + 158}{78}$$

   where x is the number of 80-byte logical records created, n is the number of records in the ISAM data set, and y is the length of a fixed record or the average length of variable records.

   Figure 77 on page 233 shows the format of an unloaded data set for the first three 100-byte records of an ISAM data set. Each is preceded by 2 bytes (bb) that indicate the number of bytes in that record. (The last record is followed by 2 bytes containing binary zeros to identify the last logical record in the unloaded data set.) The characteristics of the ISAM data set are contained in the first two logical records of the unloaded data set. Data from the ISAM data set begins in the third logical record. Each logical record in the unloaded data set contains a binary sequence number (aa) in the first 2 bytes of the record.

7. For variable records, all records in the data set must have a length equal to or greater than RKP plus KEYLEN.

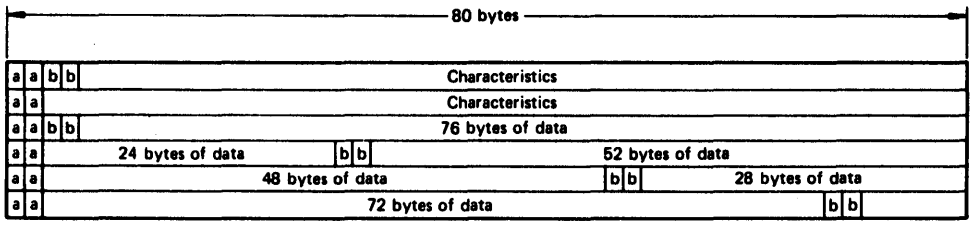


Figure 77. An Unloaded Data Set Created Using IEBISAM

CREATING AN ISAM DATA SET FROM AN UNLOADED DATA SET

An ISAM data set can be created from an unloaded version of an ISAM data set. When the unloaded data set is loaded, those records that were contained in the overflow area assigned to the original ISAM data set are moved sequentially into the primary area of the loaded ISAM data set.

PRINTING THE LOGICAL RECORDS OF AN ISAM DATA SET

The records of an ISAM data set can be printed or stored as a sequential data set for subsequent printing. Each input record is placed in a buffer from which it is printed or placed in a sequential data set. When the DELETE parameter is specified in the OPTCD field, each input record not marked for deletion is also placed in a buffer from which it is printed or placed in a sequential data set. Each printed record is converted to hexadecimal unless specified otherwise by the user.

IEBISAM provides user exits so the user can include user-written routines to:

- Modify records before printing.
- Select records for printing or terminate the printing operation after a certain number of records have been printed.
- Convert the format of a record to be printed.
- Provide a record heading for each record if the record length is at least 18 bytes.

If no user routines are provided, each record is identified in sequential order on the printout.

Exit routines must be included in either the job library or the link library.

When a user routine is supplied for a print operation, IEBISAM issues a LOAD macro instruction. A BALR 14,15 instruction is used to give control to the user's routine. When the user's routine receives control, register 0 contains a pointer to a record heading buffer; register 1 contains a pointer to an input record buffer. (The user must save registers 2 through 14 when control is given to the user routine.)

The input record buffer has a length equal to the length of the input logical record.

Figure 79 shows the record heading buffer.

The user returns control to IEBISAM by issuing a RETURN macro instruction (via register 14) or by using a BR 14 instruction after restoring registers 2 through 14.

A user routine must place a return code in register 15 before returning control to IEBISAM. The possible return codes and their meanings are listed in Figure 78.

Codes	Meaning
00 (00 hex)	Buffers are to be printed. The operation continues.
04 (04)	Buffers are to be printed. The operation is terminated. the operation is terminated.
08 (08)	This input record is not to be printed. Processing continues.
12 (0C)	This input record is not to be printed. The operation is terminated.

Figure 78. IEBISAM User Exit Return Codes

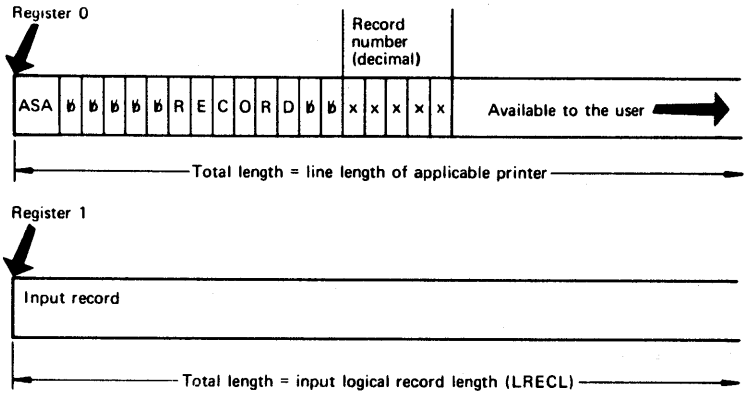


Figure 79. Record Heading Buffer Used by IEBISAM

INPUT AND OUTPUT

IEBISAM uses an input data set (the organization of the input data set depends on the operation to be performed) as follows:

- If a data set is copied, unloaded, or printed in logical sequence, the input is an ISAM data set.
- If a data set is loaded, the input is an unloaded version of an ISAM data set.

IEBISAM produces as output:

- An output data set, which is the result of the IEBISAM operation.
- A message data set, which contains information messages and any error messages.

RETURN CODES

IEBISAM returns a code in register 15 to indicate the results of program execution. The return codes and their meanings are listed below.

Codes	Meaning
00 (00 hex)	Successful completion.
04 (04)	A return code of 04 or 12 was passed to IEBISAM by the user routine.
08 (08)	An error condition occurred that caused termination of the operation.
12 (0C)	A return code other than 00, 04, 08, or 12 was passed to IEBISAM from a user routine. The job step is terminated.
16 (10)	An error condition caused termination of the operation.

Figure 80. IEBISAM Return Codes

CONTROL

IEBISAM is controlled by job control statements only. No utility control statements are required.

JOB CONTROL STATEMENTS

Figure 81 shows the job control statements for IEBISAM.

Statement	Use
JOB	Initiates the job.
EXEC	Specifies the program name (PGM=IEBISAM). Additional information is required on the EXEC statement to control the execution of IEBISAM; see "PARM Information on the EXEC Statement" below.
SYSUT1 DD	Defines the input data set.
SYSUT2 DD	Defines the output data set.
SYSPRINT DD	Defines a sequential message data set, which can be written to a system output device, a tape volume, or a direct access device.

Figure 81. Job Control Statements for IEBISAM

If the block size of the SYSPRINT data set is not a multiple of 121, a default value of 121 is taken (no error message is issued, and no condition code is set).

PARM Information on the EXEC Statement

The PARM parameter on the EXEC statement is used to control the execution of IEBISAM.

The format of the PARM parameter is:

EXEC	PARM={COPY|UNLOAD|LOAD|PRINTL[,N]} [,EXIT=routinename]

Exit routines must be included in either the job library or the link library.

For a COPY operation, the SYSUT2 DD statement must include a primary space allocation that is sufficient to accommodate records that were contained in overflow areas in the original ISAM data set. New overflow areas can be specified when the data set is copied.

For an UNLOAD operation, specifications that are implied by default or included in the DCB parameter of the SYSUT2 DD statement (for example, tape density) must be considered when the data set is subsequently loaded. If a block size is specified in the DCB parameter of the SYSUT2 DD statement, it must be a multiple of 80 bytes.

For a LOAD operation, if the input data set resides on an unlabeled tape, the SYSUT1 DD statement must specify a BLKSIZE that is a multiple of 80 bytes. Specifications that are implied by default or included in the DCB parameter of the SYSUT1 DD statement must be consistent with specifications that were implied or included in the DCB parameter of the SYSUT2 DD statement used for the UNLOAD operation. The SYSUT2 DD statement must include a primary space allocation that is sufficient to accommodate records that were contained in overflow areas in the original ISAM data set. If new overflow areas are desired, they must be specified when the data set is loaded.

For a PRINTL operation, if the device defined by the SYSUT2 DD statement is a printer, the specified BLKSIZE must be equal to or less than the physical printer size; that is 121, 133, or 145 bytes. If BLKSIZE is not specified, 121 bytes is assumed. LRECL (or BLKSIZE when no LRECL was specified) must be between 55 and 255 bytes.

If a user routine is supplied for a PRINTL operation, IEBISAM issues a LOAD macro instruction to make the user routine available. A BALR 14,15 instruction is subsequently used to give control to the routine. When the user routine receives control, register 0 contains a pointer to a record heading buffer; register 1 contains a pointer to an input record buffer.

Parameters	Applicable Control Statements	Description of Parameters
PARM	EXEC	**PARM={COPY|UNLOAD|LOAD|PRINTL[,N]} [,EXIT=routinename]** The PARM values have the following meaning: • COPY specifies a copy operation. • UNLOAD specifies an unload operation. This is the default. • LOAD specifies a load operation. • PRINTL specifies a print operation in which each record is converted to hexadecimal before printing. The N is an optional value that specifies that records are not to be converted to hexadecimal before printing. • EXIT is an optional value that specifies the name of the exit routine that is to receive control before each record is printed. See "PARM Information on the EXEC Statement" on page 236 for values that must be coded with the PARM parameter.

IEBISAM EXAMPLES

The following examples illustrate some of the uses of IEBISAM. Figure 82 can be used as a quick-reference guide to IEBISAM examples. The numbers in the "Example" column point to the examples that follow.

Operation	Data Set Organization	Device	Comments	Example
COPY	ISAM	Disks	Unblocked input; blocked output. Prime area and index separation.	1
UNLOAD	ISAM, Sequential	Disk and 9-track Tape	Blocked output.	2
UNLOAD	ISAM, Sequential	Disk and 7-track Tape	Blocked output. Data set written as second data set on input volume.	3
LOAD	Sequential, ISAM	9-track Tape and Disk	Input data set is second data set on tape volume.	4
PRINTL	ISAM, Sequential	Disk and System Printer	Blocked input. Output not converted.	5

Figure 82. IEBISAM Example Directory

Examples that use **disk** or **tape** in place of actual device numbers must be changed before use. See "DASD and Tape Device Support" on page 3 for valid device number notation.

IEBISAM EXAMPLE 1

In this example, an ISAM data set is copied from two DASD volumes. The output data is blocked.

```
//CPY       JOB  09#770,SMITH
//           EXEC PGM=IEBISAM,PARM=COPY
//SYSPRINT  DD   SYSOUT=A
//SYSUT1    DD   DSNAME=ISAM01,VOLUME=SER=(222222,333333),
//               DISP=(OLD,DELETE),UNIT=(disk,2),
//               DCB=(DSORG=IS,LRECL=500,
//                    BLKSIZE=500,RECFM=F,RKP=4)
//SYSUT2    DD   DSNAME=ISAM02(INDEX),UNIT=disk,
//               DISP=(NEW,KEEP),VOLUME=SER=444444,
//               DCB=(DSORG=IS,BLKSIZE=1000,RECFM=FB),
//               SPACE=(CYL,(2))
//           DD   DSNAME=ISAM02(PRIME),UNIT=(disk,2),
//               DCB=(DSORG=IS,BLKSIZE=1000,RECFM=FB),
//               SPACE=(CYL,(10)),
//               VOLUME=SER=(444444,555555),DISP=(NEW,KEEP)
/*
```

The job control statements are discussed below:

- EXEC specifies the program name (IEBISAM) and the COPY operation.
- SYSUT1 DD defines an ISAM input data set, ISAM01, which resides on two disk volumes.
- SYSUT2 DD defines the output data set index area, ISAM02; the index and prime areas are separated.
- The second SYSUT2 DD defines the output data set prime area. Ten cylinders are allocated for the prime area on each of the two disk volumes.

IEBISAM EXAMPLE 2

In this example, an ISAM input data set is converted into a sequential data set; the output is placed on a 9-track tape volume.

```
//STEP1     JOB  09#770,SMITH
//           EXEC PGM=IEBISAM,PARM=UNLOAD
//SYSPRINT  DD   SYSOUT=A
//SYSUT1    DD   DSNAME=INDSEQ,UNIT=disk,DISP=(OLD,KEEP),
//               VOLUME=SER=111112
//SYSUT2    DD   DSNAME=UNLDSET,UNIT=tape,LABEL=(,SL),
//               DISP=(,KEEP),VOLUME=SER=001234,
//               DCB=(RECFM=FB,LRECL=80,BLKSIZE=640)
/*
```

The job control statements are discussed below:

- EXEC specifies the program name (IEBISAM) and the UNLOAD operation.
- SYSUT1 DD defines the ISAM input data set, INDSEQ, which resides on a disk volume.

- SYSUT2 DD defines the unloaded output data set, UNLDSET. The data set consists of fixed blocked records, and is to reside as the first or only data set on a 9-track tape volume.

IEBISAM EXAMPLE 3

In this example, ISAM input is converted into a sequential data set and placed on a 7-track, tape volume.

```
//STEPA    JOB  09#770,SMITH
//         EXEC PGM=IEBISAM,PARM=UNLOAD
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=INDSEQ,UNIT=disk,DISP=(OLD,KEEP),
//              VOLUME=SER=111112
//SYSUT2   DD   DSNAME=UNLDSET,UNIT=2400-2,LABEL=(2,SL),
//              VOLUME=SER=001234,DCB=(DEN=2,
//               RECFM=FB,LRECL=80,BLKSIZE=1040,TRTCH=C),
//              DISP=(,KEEP)
/*
```

The job control statements are discussed below:

- EXEC specifies the program name (IEBISAM) and the UNLOAD operation.
- SYSUT1 DD defines the input data set, INDSEQ, which is an indexed sequential data set. The data set resides on a disk volume.
- SYSUT2 DD defines the unloaded output data set, UNLDSET. The data set consists of fixed blocked records, and is to reside as the second data set on a 7-track tape volume. The data set is written at a density of 800 bits per inch (DEN=2).

IEBISAM EXAMPLE 4

In this example, an unloaded data set is converted to the form of the original ISAM data set.

```
//STEPA    JOB  09#770,SMITH
//         EXEC PGM=IEBISAM,PARM=LOAD
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=UNLDSET,UNIT=tape,LABEL=(2,SL),
//              DISP=(OLD,KEEP),VOLUME=SER=001234
//SYSUT2   DD   DSNAME=INDSEQ,DISP=(,KEEP),DCB=(DSORG=IS),
//              SPACE=(CYL,(1)),VOLUME=SER=111112,
//              UNIT=disk
/*
```

The job control statements are discussed below:

- EXEC specifies the program name (IEBISAM) and the LOAD operation.
- SYSUT1 DD defines the input data set, UNLDSET, which is a sequential (unloaded) data set. The data set is the second data set on a tape volume.

- SYSUT2 DD defines the output data set, INDSEQ which is an ISAM data set. One cylinder of space is allocated for the data set on a disk volume.

IEBISAM EXAMPLE 5

In this example, the logical records of an ISAM data set are printed on a system output device.

```
//PRINT    JOB  09#770,SMITH
//         EXEC PGM=IEBISAM,PARM='PRINTL,N'
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=ISAM03,UNIT=disk,DISP=OLD,
//              VOLUME=SER=222222
//SYSUT2   DD   SYSOUT=A
/*
```

The job control statements are discussed below:

- EXEC specifies the program name (IEBISAM) and the PRINTL operation. The output records are not converted to hexadecimal prior to printing. (N is specified).
- SYSUT1 DD defines the input data set, ISAM03, which resides on a disk volume.
- SYSUT2 DD defines the output data set (in this case, the system printer). A logical record length (LRECL) of 121 bytes is assumed.

IEBPTPCH is a data set utility used to print or punch all, or selected portions, of a sequential or partitioned data set. Records can be printed or punched to meet either standard specifications or user specifications.

The standard specifications are:

- Each logical output record begins on a new printed line or punched card.
- Each printed line consists of groups of 8 characters separated by 2 blanks. Each punched card contains up to 80 contiguous bytes of information.
- Characters that cannot be printed appear as blanks.
- When the input is blocked, each logical output record is delimited by "*" and each block is delimited by "**."

User formats can be specified, provided that no output record exceeds the capability of the output device.

IEBPTPCH provides optional editing facilities and exits for user routines that can be used to process labels or manipulate input or output records.

IEBPTPCH can be used to print or punch:

- A sequential or partitioned data set in its entirety
- Selected members from a partitioned data set
- Selected records from a sequential or partitioned data set
- The directory of a partitioned data set
- An edited version of a sequential or partitioned data set

PRINTING OR PUNCHING AN ENTIRE DATA SET

IEBPTPCH can be used to print or punch a sequential data set or a partitioned data set in its entirety. Data to be printed or punched can be either hexadecimal or a character representation of valid alphameric bit configurations. For a print operation, packed decimal data should be converted to unpacked decimal or hexadecimal mode to ensure that all characters are printable.

For a standard print operation, each logical record is printed in groups of eight characters. Each set of eight characters is separated from the next by two blanks. Up to 96 data characters can be included on a printed line. (An edited output can be produced to omit the blank delimiters and print up to 144 characters per line.)

Data from a logical input record is punched in contiguous columns in the punched card(s) representing that record. Sequence numbers can be created and placed in columns 73 through 80 of the punched cards.

PRINTING OR PUNCHING SELECTED MEMBERS

IEBPTPCH can be used to print or punch selected members of a partitioned data set. Utility control statements are used to specify members to be printed or punched.

PRINTING OR PUNCHING SELECTED RECORDS

IEBPTPCH can be used to print or punch selected records from a sequential or partitioned data set. Utility control statements can be used to specify:

- The termination of a print or punch operation after a specified number of records has been printed or punched.
- The printing or punching of every nth record.
- The starting of a print or punch operation after a specified number of records.

PRINTING OR PUNCHING A PARTITIONED DIRECTORY

IEBPTPCH can be used to print or punch the contents of a partitioned directory. Each directory block is printed in groups of eight characters. If the directory is printed in hexadecimal representation, the first four printed characters of each directory block indicate the total number of used bytes in that block. For details of the format of the directory, see the Debugging Handbook.

Data from a directory block is punched in contiguous columns in the punched cards representing that block.

PRINTING OR PUNCHING AN EDITED DATA SET

IEBPTPCH can be used to print or punch an edited version of a sequential or a partitioned data set. Utility control statements can be used to specify editing information that applies to a record, a group of records, selected groups of records, or an entire member or data set.

An edited data set is produced by:

- Rearranging or omitting defined data fields within a record
- Converting data from packed decimal to unpacked decimal or from alphameric to hexadecimal representation

INPUT AND OUTPUT

IEBPTPCH uses the following input:

- An input data set, which contains the data that is printed or punched. The input data set can be either sequential or partitioned.
- A control data set, which contains utility control statements. The control data set is required for each use of IEBPTPCH.

IEBPTPCH produces the following output:

- An output data set, which is the printed or punched data set.
- A message data set, which contains informational messages (for example, the contents of the control statements) and any error messages.

RETURN CODES

IEBPTPCH returns a code in register 15 to indicate the results of program execution. The return codes and their meanings are listed below.

Codes	Meaning
00 (00 hex)	Successful completion.
04 (04)	Either a physical sequential data set is empty or a partitioned data set has no members.
08 (08)	A member specified for printing or punching does not exist in the input data set. Processing continues with the next member.
12 (0C)	An unrecoverable error occurred or that a user routine passed a return code of 12 to IEBPTPCH. The job step is terminated.
16 (10)	A user routine passed a return code of 16 to IEBPTPCH. The job step is terminated.

Figure 83. IEBPTPCH Return Codes

CONTROL

IEBPTPCH is controlled by job control statements and utility control statements. The job control statements are required to execute or invoke the IEBPTPCH program and to define the data sets that are used and produced by the program. The utility control statements are used to control the functions of IEBPTPCH.

JOB CONTROL STATEMENTS

Figure 84 shows the job control statements for IEBPTPCH.

Statement	Use
JOB	Initiates the job step.
EXEC	Specifies the program name (PGM=IEBPTPCH) or, if the job control statements reside in a procedure library, the procedure name.
SYSPRINT DD	Defines a sequential message data set. The data set can be written to a system output device, a tape volume, or a direct access device.
SYSUT1 DD	Defines a sequential or partitioned input data set.
SYSUT2 DD	Defines the output (print or punch) data set.
SYSIN DD	Defines the control data set. The control data set normally resides in the input stream; however, it can be defined as a member in a partitioned data set.

Figure 84. Job Control Statements for IEBPTPCH

SYSPRINT DD Statement

The SYSPRINT DD statement is required for each use of IEBPTPCH. The RECFM is always FBA, the LRECL is always 121. Output can be blocked by specifying a block size that is a multiple of 121 on the SYSPRINT DD statement. The default block size is 121.

SYSUT1 DD Statement

The SYSUT1 DD statement is required for each use of IEBPTPCH. The RECFM (except for undefined records), BLKSIZE, and LRECL (except for undefined and fixed unblocked records) must be present on the DD statement, in the DSCB, or on the tape label.

The input data set can contain fixed, variable, undefined, or variable spanned records. Variable spanned records are permitted only when the input is sequential.

A partitioned directory to be printed or punched must be defined as a sequential data set (TYPORG=PS). You must specify RECFM=U, BLKSIZE=256, and LRECL=256 on the SYSUT1 DD statement.

SYSUT2 DD Statement

The SYSUT2 DD statement is required every time IEBPTPCH is used. The RECFM is always FBA or FBM. The LRECL parameter, or, if no logical record length is specified, the BLKSIZE parameter, specifies the number of characters to be written per printed line or per punched card (this count includes a control character). The number of characters specified must be in the range of 2 through 145. The default values for edited output lines are 121 characters per printed line and 81 characters per punched card.

The SYSUT2 data set can be blocked by specifying both the LRECL and the BLKSIZE parameters, in which case, block size must be a multiple of logical record length.

Both the output data set and the message data set can be written to the system output device if it is a printer.

If the logical record length of the input records is such that the output would exceed the output record length, IEBPTPCH divides the record into multiple lines or cards in the case of standard printed output, standard punched output, or when the PREFORM parameter is specified. For nonstandard output, or if the PREFORM parameter is not specified, only part of the input record is printed or punched (maximums determined by the specific characteristics of your output device).

SYSIN DD Statement

The SYSIN DD statement is required for each use of IEBPTPCH. The RECFM is always FB, the LRECL is always 80. Any blocking factor that is a multiple of 80 can be specified for the BLKSIZE. The default block size is 80.

UTILITY CONTROL STATEMENTS

IEBPTPCH is controlled by utility control statements. The control statements in Figure 85 on page 245 are shown in the order in which they must appear.

Control statements are included in the control data set, as required. Any number of MEMBER and RECORD statements can be included in a job step.

A nonblank character in column 72 is optional for IEBPTPCH continuation statements. Continuation requirements for utility control statements are described in "Continuing Utility Control Statements" on page 5.

Statement	Use
PRINT	Specifies that the data is printed.
PUNCH	Specifies that the data is punched.
TITLE	Specifies that a title is to precede the printed or punched data.
EXITS	Specifies that user exit routines are provided.
MEMBER	Specifies that the input is a partitioned data set and that a selected member is printed or punched.
RECORD	Specifies whether editing is performed, that is, records are to be printed or punched to the user's specifications.
LABELS	Specifies whether user labels are treated as data.

Figure 85. IEBPTPCH Utility Control Statements

PRINT Statement

The PRINT statement is used to initiate the IEBPTPCH PRINT operation. If used, PRINT must be the first statement in the control data set.

The format of the PRINT statement is:

[label]	**PRINT**	**[PREFORM=A|M]** **[,TYPORG=PS|PO]** **[,TOTCONV=XE|PZ]** **[,CNTRL=n|1]** **[,STRTAFT=n]** **[,STOPAFT=n]** **[,SKIP=n]** **[,MAXNAME=n]** **[,MAXFLDS=n]** **[,MAXGPS=n]** **[,MAXLITS=n]** **[,INITPG=n]** **[,MAXLINE=n]**

PUNCH Statement

The PUNCH statement is used to initiate the IEBPTPCH PUNCH operation. If used, PUNCH must be the first statement in the control data set.

The format of the PUNCH statement is:

[label]	PUNCH	[PREFORM=A|M] [,TYPORG=PS|PO] [,TOTCONV=XE|PZ] [,CNTRL=n|1] [,STRTAFT=n] [,STOPAFT=n] [,SKIP=n] [,MAXNAME=n] [,MAXFLDS=n] [,MAXGPS=n] [,MAXLITS=n] [,CDSEQ=n] [,CDINCR=n]

TITLE Statement

The TITLE statement is used to request title and subtitle records. Two TITLE statements can be included for each use of IEBPTPCH. A first TITLE statement defines the title, and a second defines the subtitle. The TITLE statement, if included, follows the PRINT or PUNCH statement in the control data set.

The format of the TITLE statement is:

[label]	TITLE	ITEM=('title'[,output-location])

The literal coded for 'title' is not affected by the TOTCONV parameter.

EXITS Statement

The EXITS statement is used to identify exit routines supplied by the user. Exits to label processing routines are ignored if the input data set is partitioned. Linkage to and from user routines are discussed in Appendix A, "Exit Routine Linkage" on page 438.

The EXITS statement, if included, must immediately follow any TITLE statement or follow the PRINT or PUNCH statement.

The format of the EXITS statement is:

[label]	**EXITS**	**[INHDR=**routinename**]** **[,INTLR=**routinename**]** **[,INREC=**routinename**]** **[,OUTREC=**routinename**]**

MEMBER Statement

The MEMBER statement is used to identify members to be printed or punched. All RECORD statements that follow a MEMBER statement pertain to the member indicated in that MEMBER statement only. When RECORD and MEMBER statements are used, at least one MEMBER statement must precede the first RECORD statement. If no RECORD statement is used, the member is processed to standard specifications.

If no MEMBER statement appears, and a partitioned data set is being processed, all members of the data set are printed or punched. Any number of MEMBER statements can be included in a job step.

If a MEMBER statement is present in the input stream, MAXNAME must be specified in a PRINT or PUNCH statement.

The format of the MEMBER statement is:

[label]	**MEMBER**	**NAME={**membername**|**aliasname**}**

RECORD Statement

The RECORD statement is used to define a group of records, called a **record group**, that is printed or punched to the user's specifications. A record group consists of any number of records to be edited identically.

If no RECORD statements appear, the entire data set, or named member, is printed or punched to standard specifications. If a RECORD statement is used, all data following the record group it defines (within a partitioned member or within an entire sequential data set) must be defined with other RECORD statements. Any number of RECORD statements can be included in a job step.

A RECORD statement referring to a partitioned data set for which no members have been named need contain only FIELD parameters. These are applied to the records in all members of the data set.

If a FIELD parameter is included in the RECORD statement, MAXFLDS must be specified in the PRINT or PUNCH statement.

If an IDENT parameter is included in the RECORD statement, MAXGPS and MAXLITS must be specified in the PRINT or PUNCH statement.

The format of the RECORD statement is:

[label]	RECORD	[IDENT=(length,'name',input-location)] [,FIELD=(length [,input-location] [,conversion] [,output-location])]

LABELS Statement

The LABELS statement specifies whether user labels are treated as data. For a detailed discussion of this option, refer to Appendix C, "Processing User Labels" on page 446.

LABELS DATA=NO must be specified to make standard user label (SUL) exits inactive when an input data set with nonstandard labels (NSL) is processed.

If more than one valid LABELS statement is included, all but the last LABELS statement are ignored.

The format of the LABELS statement is:

[label]	LABELS	[CONV=PZ|XE] [,DATA=YES|NO|ALL|ONLY]

Parameters	Applicable Control Statements	Description of Parameters
CDINCR	PUNCH	**CDINCR=**n specifies the increment to be used in generating sequence numbers. **Default:** 10 is the increment value.
CDSEQ	PUNCH	**CDSEQ=**n specifies the initial sequence number of a deck of punched cards. This value must be contained in columns 73 through 80. Sequence numbering is initialized for each member of a partitioned data set. If the value of n is zero, 00000000 is the starting sequence number. **Default:** Cards are not numbered.
CNTRL	PRINT	**CNTRL=**n|**1** specifies a control character for the output device that indicates line spacing, as follows: 1 indicates single spacing (the default), 2 indicates double spacing, and 3 indicates triple spacing.
	PUNCH	specifies a control character for the output device that is used to select the stacker, as follows: 1 indicates the first stacker (the default), 2 indicates the second stacker, and 3 indicates the third stacker, if any.
CONV	LABELS	**CONV=PZ|XE** specifies a 2-byte code that indicates the type of conversion to be performed on this field before it is printed or punched. The values that can be coded are: **PZ** specifies that data (packed decimal) is converted to unpacked decimal data. The converted portion of the input record (length L) occupies 2L - 1 output characters when punching, and 2L output characters when printing. **XE** specifies that data (alphameric) is converted to hexadecimal data. The converted portion of the input record (length L) occupies 2L output characters. **Default:** The field is moved to the output area without change.

Parameters	Applicable Control Statements	Description of Parameters
DATA	LABELS	**DATA=YES|NO|ALL|ONLY** specifies whether user labels are treated as data. The values that can be coded are: **YES** specifies that any user labels that are not rejected by a user's label processing routine are treated as data. Processing of labels as data stops in compliance with standard return codes. YES is the default. **NO** specifies that user labels are not to be treated as data. NO must be specified when processing input/output data sets with nonstandard labels (NSL) in order to make standard user label (SUL) exits inactive. **ALL** specifies that all user labels are treated as data. A return code of 16 causes the utility to complete the processing of the remainder of the group of user labels and to terminate the job step. **ONLY** specifies that only user header labels are treated as data. User header labels are processed as data regardless of any return code. The job terminates upon return from the OPEN routine.

Parameters	Applicable Control Statements	Description of Parameters
FIELD	RECORD	**FIELD=(**length**,[**input-location**],[**conversion**],[**output-location**])[,FIELD=....]** specifies field-processing and editing information. Note that the variables on the FIELD parameter are positional; that is, if any of the options are not coded, the associated comma preceding that variable must be coded. These values can be coded: length specifies the length (in bytes) of the input field to be processed. The length must be equal to or less than the initial input LRECL. input-location specifies the starting byte of the input field to be processed. The sum of the length and the input location must be equal to or less than the input LRECL plus one. **Default:** Byte 1 is assumed. conversion specifies a 2-byte code that indicates the type of conversion to be performed on this field before it is printed or punched. The values that can be coded are: **PZ** specifies that data (packed decimal) is converted to unpacked decimal data. The converted portion of the input record (length L) occupies 2L - 1 output characters when punching, and 2L output characters when printing. **XE** specifies that data (alphameric) is converted to hexadecimal data. The converted portion of the input record (length L) occupies 2L output characters. **Default:** The field is moved to the output area without change.

Parameters	Applicable Control Statements	Description of Parameters
FIELD (continued)	RECORD	output-location specifies the starting location of this field in the output records. Unspecified fields in the output records appear as blanks in the printed or punched output. Data that exceeds the SYSUT2 printer or punch size is not printed or punched. The specified fields may not exceed the logical output record length minus one. When specifying one or more FIELDs, the sum of all lengths and all extra characters needed for conversions must be equal to or less than the output LRECL minus one. **Default:** Byte 1 is assumed. If a FIELD parameter is included in the RECORD statement, MAXFLDS must be specified in the PRINT or PUNCH statement.
IDENT	RECORD	**IDENT=**(length,'name',input-location) identifies the last record of the record group to which the FIELD parameters apply. The values that can be coded are: length specifies the length (in bytes) of the field that contains the identifying name in the input records. The length cannot exceed 8 bytes. 'name' specifies the exact literal, enclosed in apostrophes, that identifies the last record of a record group. If the literal contains apostrophes, each must be written as two consecutive apostrophes. input-location specifies the starting location of the field that contains the identifying name in the input records. The sum of the length and the input location must be equal to or less than the input LRECL plus one. **Default:** If IDENT is omitted and STOPAFT is not included with the PRINT or PUNCH statement, record processing halts after the last record in the data set. If IDENT is omitted and STOPAFT is included with the PRINT or PUNCH statement, record processing halts when the STOPAFT count is satisfied or after the last record of the data set is processed, whichever occurs first. If an IDENT parameter is included in the RECORD statement, MAXGPS and MAXLITS must be specified in the PRINT or PUNCH statement.

Parameters	Applicable Control Statements	Description of Parameters
INHDR	EXITS	**INHDR=**routinename specifies the name of the routine that processes user input header labels.
INITPG	PRINT	**INITPG=**n specifies the initial page number; the pages are numbered sequentially thereafter. The INITPG parameter must not exceed a value of 9999. **Default:** Page 1
INREC	EXITS	**INREC=**routinename specifies the name of the routine that manipulates each logical record (or physical block in the case of VS or VBS records longer than 32K bytes) before it is processed.
INTLR	EXITS	**INTLR=**routinename specifies the name of the routine that processes user input trailer labels.
ITEM	TITLE	**ITEM=('**title**'[,**output-location**])[,ITEM...]** specifies title or subtitle information. The values that can be coded are: 'title' specifies the title or subtitle literal (maximum length of 40 bytes), enclosed in apostrophes. If the literal contains apostrophes, each apostrophe must be written as two consecutive apostrophes. output-location specifies the starting position at which the literal for this item is placed in the output record. When used with output-location, the specified title's length plus output-location may not exceed the output logical record length minus one. **Default:** Byte 1 is assumed.
MAXFLDS	PRINT PUNCH	**MAXFLDS=**n specifies a number no less than the total number of FIELD parameters appearing in subsequent RECORD statements. The value must not exceed 32767. If MAXFLDS is omitted when there is a FIELD parameter present, the print or punch request is terminated.

Parameters	Applicable Control Statements	Description of Parameters
MAXGPS	PRINT PUNCH	**MAXGPS=**n specifies a number no less than the total number of IDENT parameters appearing in subsequent RECORD statements. The value must not exceed 32767. If MAXGPS is omitted when there is an IDENT parameter present, the print or punch request is terminated.
MAXLINE	PRINT	**MAXLINE=**n specifies the maximum number of lines to a printed page. Spaces, titles, and subtitles are included in this number. **Default:** 60 lines per page.
MAXLITS	PRINT PUNCH	**MAXLITS=**n specifies a number no less than the total number of characters contained in the IDENT literals of subsequent RECORD statements. The value must not exceed 32767. If MAXLITS is omitted when there is a literal present, the print or punch request is terminated.
MAXNAME	PRINT PUNCH	**MAXNAME=**n specifies a number no less than the total number of member names and aliases appearing in subsequent MEMBER statements. The value must not exceed 32767. If MAXNAME is omitted when there is a MEMBER statement present, the print or punch request is terminated.
NAME	MEMBER	**NAME=**{membername|aliasname} specifies a member to be printed or punched. The values that can be coded are: membername specifies a member by its member name. aliasname specifies a member by its alias name. If a MEMBER statement is present in the input stream, MAXNAME must be specified in a PRINT or PUNCH statement.
OUTREC	EXITS	**OUTREC=**routinename specifies the name of the routine that manipulates each logical record (or physical block in the case of VS or VBS records longer than 32K bytes) before it is printed or punched.

Parameters	Applicable Control Statements	Description of Parameters
PREFORM	PRINT PUNCH	**PREFORM=A|M** specifies that a control character is provided as the first character of each record to be printed or punched. The control characters are used to control the spacing, number of lines per page, page ejection, and selecting a stacker. That is, the output has been previously formatted, and the "standard specifications" are superseded. If an error occurs, the print/punch operation is terminated. If PREFORM is coded, any additional PRINT or PUNCH operands and all other control statements, except for syntax checking, LABELS statements and TYPORG operands, are ignored. PREFORM must not be used for printing or punching data sets with VS or VBS records longer than 32K bytes. These values are coded as follows: **A** specifies that an ASA control character is provided as the first character of each record to be printed or punched. If the input record length exceeds the output record length, the utility uses the ASA character for printing the first line, with a single space character on all subsequent lines of the record (for PRINT), or duplicates the ASA character on each output card of the record (for PUNCH). **M** specifies that a machine-code control character is provided as the first character of each record to be printed or punched. If the input record length exceeds the output record length, the utility prints all lines of the record with a print-skip-one-line character until the last line of the record, which will contain the actual character provided as input (for PRINT), or duplicates the machine control character on each output card of the record (for PUNCH).
SKIP	PRINT PUNCH	**SKIP=n** specifies that every nth record (or physical block in the case of VS or VBS records longer than 32K bytes) is printed or punched. **Default:** Successive logical records are printed or punched.

Parameters	Applicable Control Statements	Description of Parameters
STOPAFT	PRINT PUNCH	**STOPAFT=n** specifies, for sequential data sets, the number of logical records (or physical blocks in the case of VS or VBS records longer than 32K bytes) to be printed or punched. For partitioned data sets, this specifies the number of logical records (or physical blocks in the case of VS or VBS records longer than 32K bytes) to be printed or punched in each member to be processed. The n value must not exceed 32767. If STOPAFT is specified and the IDENT parameter of the RECORD statement is also specified, the operation is terminated when the STOPAFT count is satisfied or at the end of the first record group, whichever occurs first.
STRTAFT	PRINT PUNCH	**STRTAFT=n** specifies, for sequential data sets, the number of logical records (physical blocks in the case of variable spanned (VS) or variable block spanned (VBS) type records longer than 32K bytes) to be skipped before printing or punching begins. For partitioned data sets, STRTAFT=n specifies the number of logical records to be skipped in each member before printing or punching begins. The n value must not exceed 32767. If STRTAFT is specified and RECORD statements are present, the first RECORD statement of a member describes the format of the first logical record to be printed or punched.

Parameters	Applicable Control Statements	Description of Parameters
TOTCONV	PRINT PUNCH	**TOTCONV=XE|PZ** specifies the representation of data to be printed or punched. TOTCONV can be overridden by any user specifications (RECORD statements) that pertain to the same data. These values are coded as follows: **XE** specifies that data is punched in 2-character-per-byte hexadecimal representation (for example, C3 40 F4 F6). If XE is not specified, data is punched in 1-character per byte alphameric representation. The above example would appear as C 46. The converted portion of the input record (length L) occupies 2L output characters. **PZ** specifies that data (packed decimal mode) is converted to unpacked decimal mode. IEBPTPCH does not check for packed decimal mode. The converted portion of the input record (length L) occupies 2L-1 output characters when punching, and 2L output characters **Default:** If TOTCONV is omitted, data is not converted.
TYPORG	PRINT PUNCH	**TYPORG=PS|PO** specifies the organization of the input data set. These values are coded as follows: **PS** specifies that the input data set is organized sequentially. This is the default. **PO** specifies that the input data set is partitioned.

IEBPTPCH EXAMPLES

The following examples illustrate some of the uses of IEBPTPCH. Figure 86 can be used as a quick-reference guide to IEBPTPCH examples. The numbers in the "Example" column refer to the examples that follow:

Operation	Data Set Organization	Devices	Comments	Example
PRINT	Sequential	9-track Tape and System Printer	Standard format. Conversion to hexadecimal.	1
PUNCH	Sequential	7-track Tape and Card Reader	Standard format. Conversion to hexadecimal.	2
PRINT	Partitioned	Disk and System Printer	Standard format. Conversion to hexadecimal. Ten records from each member are printed.	3
PRINT	Partitioned	Disk and System Printer	Standard format. Conversion to hexadecimal. Two members are printed.	4
PRINT	Sequential	9-track Tape and System Printer	User-specified format. Input data set is the second data set on the volume.	5
PUNCH	Sequential	Disk and Card Reader Punch	User-specified format. Sequence numbers are assigned and punched.	6
PRINT	Sequential, Partitioned	Disk and System Printer	Standard format. Conversion to hexadecimal.	7
PUNCH	Sequential	Card Reader and Card Read Punch	Standard format. Control data set is a member in a cataloged partitioned data set.	8
PRINT	Sequential	Disk and System Printer	User-specified format. User routines are provided. Processing ends after the third record group is printed or STOPAFT is satisfied.	9
PRINT	Sequential	9-track Tape and System Printer	SYSOUT format. SYSOUT data set is on tape volume.	10

Figure 86. IEBPTPCH Example Directory

Examples that use **disk** or **tape** in place of actual device numbers must be changed before use. See "DASD and Tape Device Support" on page 3 for valid device number notation.

IEBPTPCH EXAMPLE 1

In this example, a sequential data set is printed according to standard specifications. The printed output is converted to hexadecimal.

```
//PRINT    JOB  09#660,SMITH
//         EXEC PGM=IEBPTPCH
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   UNIT=tape,LABEL=(,NL),VOLUME=SER=001234,
//              DISP=(OLD,KEEP),DCB=(RECFM=U,BLKSIZE=2000)
//SYSUT2   DD   SYSOUT=A
//SYSIN    DD   *
     PRINT   TOTCONV=XE
     TITLE   ITEM=('PRINT SEQ DATA SET WITH CONV TO HEX',10)
/*
```

The control statements are discussed below.

- SYSUT1 DD defines the input data set on a tape volume. The data set contains undefined records; no record is larger than 2,000 bytes.
- SYSUT2 DD defines the output data set. The data set is written to the system output device (printer assumed). Each printed line contains groups (8 characters each) of hexadecimal information. Each input record begins a new line of printed output. The size of the input record and the carriage width determine how many lines of printed output are required per input record.
- SYSIN DD defines the control data set, which follows in the input stream. The control data set contains the PRINT and TITLE statements.
- PRINT initiates the print operation and specifies conversion from alphameric to hexadecimal representation.
- TITLE specifies a title to be placed beginning in column 10 of the printed output. The title is not converted to hexadecimal.

IEBPTPCH EXAMPLE 2

In this example, a sequential data set is punched according to standard specifications. The punched output is converted to hexadecimal.

```
//PUNCHSET JOB  09#660,SMITH
//         EXEC PGM=IEBPTPCH
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=INSET,UNIT=tape,VOLUME=SER=001234,
//              LABEL=(,NL),DISP=(OLD,KEEP),DCB=(RECFM=FB,
//              LRECL=80,BLKSIZE=2000)
//SYSUT2   DD   SYSOUT=B
//SYSIN    DD   *
     PUNCH   TOTCONV=XE
     TITLE   ITEM=('PUNCH SEQ DATA SET WITH CONV TO HEX',10)
/*
```

The control statements are discussed below:

- SYSUT1 DD defines the input data set, called INSET, on a tape volume. The data set contains 80-byte, fixed blocked records.
- SYSUT2 DD defines the system output class (punch is assumed). Each record from the input data set is represented by two punched cards.

- SYSIN DD defines the control data set, which follows in the input stream. The control data set contains the PUNCH and TITLE statements.
- PUNCH initiates the punch operation and specifies conversion from alphameric to hexadecimal representation.
- TITLE specifies a title to be placed beginning in column 10. The title is not converted to hexadecimal.

IEBPTPCH EXAMPLE 3

In this example, a partitioned data set (ten records from each member) is printed according to standard specifications. The printed output is converted to hexadecimal.

```
//PRINTPDS JOB  09#660,SMITH
//         EXEC PGM=IEBPTPCH
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=PDS,UNIT=disk,DISP=(OLD,KEEP),
//              VOLUME=SER=111112
//SYSUT2   DD   SYSOUT=A
//SYSIN    DD   *
         PRINT  TOTCONV=XE,TYPORG=PO,STOPAFT=10
         TITLE  ITEM=('PRINT PDS - 10 RECS EACH MEM',20)
/*
```

The control statements are discussed below:

- SYSUT1 DD defines the input data set, called PDS, on a disk volume.
- SYSUT2 DD defines the output data set on the system output device (printer assumed). Each printed line contains groups (8 characters each) of hexadecimal information. Each input record begins a new line of printed output. The size of the input record and the carriage width determine how many lines of printed output are required per input record.
- SYSIN DD defines the control data set, which follows in the input stream. The control data set contains the PRINT and TITLE statements.
- PRINT initiates the print operation, specifies conversion from alphameric to hexadecimal representation, indicates that the input data set is partitioned, and specifies that 10 records from each member are to be printed.
- TITLE specifies a title to be placed beginning in column 20 of the printed output. The title is not converted to hexadecimal.

IEBPTPCH EXAMPLE 4

In this example, two partitioned members are printed according to standard specifications. The printed output is converted to hexadecimal.

```
//PRNTMEMS JOB  09#660,SMITH
//         EXEC PGM=IEBPTPCH
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=PDS,DISP=(OLD,KEEP),VOLUME=SER=111112,
//              UNIT=disk
//SYSUT2   DD   SYSOUT=A
//SYSIN    DD   *
        PRINT   TYPORG=PO,TOTCONV=XE,MAXNAME=2
        TITLE   ITEM=('PRINT TWO MEMBS WITH CONV TO HEX',10)
       MEMBER   NAME=MEMBER1
       MEMBER   NAME=MEMBER2
/*
```

The control statements are discussed below:

- SYSUT1 DD defines the input data set, called PDS, on a disk volume.
- SYSUT2 DD defines the output data set on the system output device (printer assumed). Each printed line contains groups (8 characters each) of hexadecimal information. Each input record begins a new line of printed output. The size of the input record and the carriage width determine how many lines of printed output are required per input record.
- SYSIN DD defines the control data set, which follows in the input stream. The control data set contains PRINT, TITLE, and MEMBER statements.
- PRINT initiates the print operation, indicates that the input data set is partitioned, specifies conversion from alphameric to hexadecimal representation, and indicates that two MEMBER statements appear in the control data set (MAXNAME=2).
- TITLE specifies a title to be placed beginning in column 10 of the printed output. The title is not converted to hexadecimal.
- MEMBER specifies the member names of the members to be printed (MEMBER1 and MEMBER2).

IEBPTPCH EXAMPLE 5

In this example, a sequential data set is printed according to user specifications.

```
//PTNONSTD JOB  09#660,SMITH
//         EXEC PGM=IEBPTPCH
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=SEQSET,UNIT=tape,LABEL=(2,SUL),
//              DISP=(OLD,KEEP),VOLUME=SER=001234
//SYSUT2   DD   SYSOUT=A
//SYSIN    DD   *
        PRINT   MAXFLDS=1
        EXITS   INHDR=HDRIN,INTLR=TRLIN
       RECORD   FIELD=(80)
       LABELS   DATA=YES
/*
```

The control statements are discussed below:

- SYSUT1 DD defines the input data set, called SEQSET, which is the second data set on a tape volume.
- SYSUT2 DD defines the output data set on the system output device (printer assumed). Each printed line contains 80 contiguous characters (one record) of information.
- SYSIN DD defines the control data set, which follows in the input stream. The control data set contains the PRINT, EXITS, RECORD, and LABELS statements.
- PRINT initiates the print operation and indicates that one FIELD parameter is included in a subsequent RECORD statement (MAXFLDS=1).
- EXITS indicates that exits will be taken to user header label and trailer label processing routines when these labels are encountered on the SYSUT1 data set.
- RECORD indicates that each input record is processed in its entirety (80 bytes). Each input record is printed in columns 1 through 80 on the printer.
- LABELS specifies that user header and trailer labels are printed according to the return code issued by the user exits.

IEBPTPCH EXAMPLE 6

In this example, a sequential data set is punched according to user specifications.

```
//PHSEQNO  JOB  09#660,SMITH
//         EXEC PGM=IEBPTPCH
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=SEQSET,UNIT=disk,LABEL=(,SUL),
//              VOLUME=SER=111112,DISP=(OLD,KEEP),
//              DCB=(RECFM=FB,LRECL=80,BLKSIZE=2000)
//SYSUT2   DD   SYSOUT=B
//SYSIN    DD   *
       PUNCH   MAXFLDS=1,CDSEQ=00000000,CDINCR=20
      RECORD   FIELD=(72)
      LABELS   DATA=YES
/*
```

The control statements are discussed below:

- SYSUT1 DD defines the input data set, called SEQSET, which resides on a disk volume. The data set contains 80-byte, fixed blocked records.
- SYSUT2 DD defines the system output class (punch is assumed). That portion of each record from the input data set defined by the FIELD parameter is represented by one punched card.
- SYSIN DD defines the control data set, which follows in the input stream. The control data set contains the PUNCH, RECORD, and LABELS statements.
- PUNCH initiates the punch operation, indicates that one FIELD parameter is included in a subsequent RECORD statement (MAXFLDS=1), and assigns a sequence number for the first punched card (00000000) and an increment value for successive sequence numbers (20). Sequence numbers are placed in columns 73 through 80 of the output records.

- RECORD indicates that bytes 1 through 72 of the input records are to be punched. Bytes 73 through 80 of the input records are replaced by the new sequence numbers in the output card deck.
- LABELS specifies that user header labels and user trailer labels are punched.

Labels cannot be edited; they are always moved to the first 80 bytes of the output buffer. No sequence numbers are present on the cards containing user header and user trailer records.

IEBPTPCH EXAMPLE 7

In this example, the directory of a partitioned data set is printed. The printed output is converted to hexadecimal.

```
//PRINTDIR JOB  09#660,SMITH
//         EXEC PGM=IEBPTPCH
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=PDS,UNIT=disk,VOLUME=SER=111112,
//              DISP=(OLD,KEEP),DCB=(RECFM=U,BLKSIZE=256)
//SYSUT2   DD   SYSOUT=A
//SYSIN    DD   *
   PRINT   TYPORG=PS,TOTCONV=XE
   TITLE   ITEM=('PRINT PARTITIONED DIRECTORY OF PDS',10)
   TITLE   ITEM=('FIRST TWO BYTES SHOW NUM OF USED BYTES',10)
  LABELS   DATA=NO
/*
```

The control statements are discussed below:

- SYSUT1 DD defines the input data set (the partitioned directory), which resides on a disk volume.
- SYSUT2 DD defines the output data set on the system output device (printer assumed). Each printed line contains groups (8 characters each) of hexadecimal information. Each input record begins a new line of printed output. The size of the input record and the carriage width determine how many lines of printed output are required per input record.
- SYSIN DD defines the control data set, which follows in the input stream. The data set contains the PRINT, TITLE, and LABELS statements.
- PRINT initiates the print operation, indicates that the partitioned directory is organized sequentially, and specifies conversion from alphameric to hexadecimal representation.
- The first TITLE statement specifies a title, which is not converted to hexadecimal.
- The second TITLE statement specifies a subtitle, which is also not converted to hexadecimal.
- LABELS specifies that no user labels are printed.

Note: Not all of the bytes in a directory block need contain data pertaining to the partitioned data set; unused bytes are sometimes used by the operating system as temporary work areas. With conversion to hexadecimal representation, the first four characters of printed output indicate how many bytes of the 256-byte block pertain to the partitioned data set. Any unused bytes occur in the latter portion of the directory block; they are not interspersed with the used bytes.

IEBPTPCH EXAMPLE 8

In this example, a card deck containing valid punch card code or BCD is duplicated.

```
//PUNCH    JOB  09#660,SMITH
//         EXEC PGM=IEBPTPCH
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   DSNAME=PDSLIB(PNCHSTMT),DISP=(OLD,KEEP)
//SYSUT2   DD   SYSOUT=B
//SYSUT1   DD   DATA

(input card data set including // cards)
/*
```

The control statements are discussed below:

- SYSIN DD defines the control data set. The control data set contains a PUNCH statement and is defined as a member of the partitioned data set PDSLIB. (The data set is cataloged.) The RECFM must be FB and the LRECL must be 80.
- SYSUT2 DD defines the system output class (punch is assumed).
- SYSUT1 DD defines the input card data set, which follows in the input stream.

IEBPTPCH EXAMPLE 9

In this example, three record groups are printed. A user routine is provided to manipulate output records before they are printed.

```
                                                                      72
//PRINT    JOB  09#660,SMITH
//         EXEC PGM=IEBPTPCH
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=SEQDS,UNIT=disk,DISP=(OLD,KEEP),
//              LABEL=(,SUL),VOLUME=SER=111112
//SYSUT2   DD   SYSOUT=A
//SYSIN    DD   *
       PRINT    MAXFLDS=9,MAXGPS=9,MAXLITS=23,STOPAFT=32767
       TITLE    ITEM=('TIMECONV-DEPT D06'),                          C
                ITEM=('JAN10-17',80)
       EXITS    OUTREC=NEWTIME,INHDR=HDRS,INTLR=TLRS
       RECORD   IDENT=(6,'498414',1),                                C
                FIELD=(8,1,,10),FIELD=(30,9,XE,20)
       RECORD   IDENT=(2,'**',39),                                   C
                FIELD=(8,1,,10),FIELD=(30,9,XE,20)
       RECORD   IDENT=(6,'498414',1),                                C
                FIELD=(8,1,,10),FIELD=(30,9,XE,20)
       LABELS   CONV=XE,DATA=ALL
/*
```

The control statements are discussed below:

- SYSUT1 DD defines the input data set, called SEQDS. The data set resides on a disk volume.
- SYSUT2 DD defines the output data set on the system output device (printer assumed).

- SYSIN DD defines the control data set, which follows in the input stream. The control data set contains the PRINT, TITLE, EXITS, RECORD, and LABELS statements.
- The PRINT statement:
  1. Initializes the print operation.
  2. Indicates that not more than nine FIELD parameters are included in subsequent RECORD statements (MAXFLDS=9).
  3. Indicates that not more than nine IDENT parameters are included in subsequent RECORD statements (MAXGPS=9).
  4. Indicates that not more than 23 literal characters are included in subsequent IDENT parameters (MAXLITS=23).
  5. Indicates that processing is terminated after 32767 records are processed or after the third record group is processed, whichever comes first. Because MAXLINE is omitted, 60 lines are printed on each page.
- TITLE specifies two titles, to be printed on one line. The titles are not converted to hexadecimal.
- EXITS specifies the name of a user routine (NEWTIME), which is used to manipulate output records before they are printed.
- The first RECORD statement defines the first record group to be processed and indicates where information from the input records is placed in the output records. Bytes 1 through 8 of the input records appear in columns 10 through 17 of the printed output, and bytes 9 through 38 are printed in hexadecimal representation and placed in columns 20 through 79.
- The second RECORD statement defines the second group to be processed. The parameter in the IDENT operand specifies that an input record containing the two characters ** in positions 39 and 40 is the last record edited according to the FIELD operand in this RECORD statement. The FIELD operand specifies that bytes 1 through 8 of the input records are placed in columns 10 through 17 of the printed output, and bytes 9 through 38 are printed in hexadecimal representation and appear in columns 20 through 79.
- The third and last RECORD statement is equal to the first RECORD statement. An input record that meets the parameter in the IDENT operand ends processing, unless the STOPAFT parameter in the PRINT statement has not already done so.
- LABELS specifies that all user header or trailer labels are to be printed regardless of any return code, except 16, issued by the user's exit routine. It also indicates that the labels are converted from alphameric to hexadecimal representation (CONV=XE).

IEBPTPCH EXAMPLE 10

In this example, the input is a SYSOUT (sequential) data set, which was previously written as the second data set of a standard label tape. It is printed in SYSOUT format.

```
//PTSYSOUT JOB  09#660,SMITH
//         EXEC PGM=IEBPTPCH
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   UNIT=tape,LABEL=(2,SL),DSNAME=LISTING,
//              DISP=(OLD,KEEP),VOL=SER=001234
//SYSUT2   DD   SYSOUT=A
//SYSIN    DD   *
    PRINT   PREFORM=A
/*
```

The control statements are discussed below:

- SYSUT1 DD defines the input data set, which was previously written as the second data set of a standard label tape. The data set has been assigned the name LISTING.
- SYSUT2 DD defines the output data set on the system output device (printer assumed).
- SYSIN DD defines the control data set, which follows in the input stream. The control data set contains the PRINT statement.
- The PRINT statement initiates the print operation and indicates that an ASA control character is provided as the first character of each record to be printed (PREFORM=A).

IEBTCRIN PROGRAM

IEBTCRIN is a data set utility used to read input from the IBM 2495 Tape Cartridge Reader (TCR), edit the data as specified by the user, and produce a sequentially organized output data set.

IEBTCRIN can be used to construct records from the stream of data bytes read sequentially from the Tape Cartridge Reader. The user has the option of gaining temporary control (via a user-supplied exit routine) to process each logical record.

The input to IEBTCRIN is in the form of cartridges written by either the IBM Magnetic Tape Selectric Typewriter (MTST) or the IBM 50 Magnetic Data Inscriber (MTDI). An input data set (one or more cartridges) must consist of either all MTST cartridges or all MTDI cartridges. (For more information concerning the MTDI use and an explanation of terminology used in this chapter, refer to IBM 50 Magnetic Data Inscriber Component Description.)

When MTDI input is edited, IEBTCRIN maintains information about each record as it is being edited. This information is summarized in "Error Description Word (EDW)" on page 274. When the EDW contains a value other than zero in either the level status (byte 0) or the type status (byte 1), the record is considered an error record by the program and the EDW is added to the start of the record to aid the user in analyzing the error.

MTDI EDITING CRITERIA

The cartridges created on the IBM 50 Magnetic Data Inscriber contain a continuous stream of data bytes (that is, there are no interblock gaps). Therefore, when editing is specified, IEBTCRIN extracts records one at a time from the data stream. To accomplish this, IEBTCRIN scans for control codes written by MTDI. IEBTCRIN uses start-of-record (SOR) and end-of-record (EOR) locations to extract MTDI records from the input stream.

The (SOR) location is defined as:

- The location of the first character on a cartridge.
- The location of the first character after the previous record's (EOR) location.
- The location of an SOR code.
- The location of a group separator (GS) code.

The character in the SOR location is checked to determine if it is a valid start-of-record character. A P1 through P8, a cancel code, or a GS code are valid start-of-record characters; all others are invalid.

The EOR location by priority sequence is:

1. The same location as the SOR location, if the SOR character was a valid GS code.
2. The location of the first encountered record mark (RM) or verify okay (VOK) code if that location is within the length of the maximum user-specified record size.
3. The location of any code preceding either a valid SOR code or the end-of-media (EOM) code, if that location is within the length of the maximum user-specified record size.

4. The location determined in 2 or 3, regardless of the maximum user-specified record size if the SOR location contains a cancel code.

5. If one of the previous EOR locations cannot be defined, an EOR condition will be forced at the location where the record length equals the maximum user-specified record size.

The character in the EOR location is checked to determine if it is a valid end-of-record character. Valid EOR characters are the GS character (if the SOR character was a GS code) and VOK or RM codes; all others are invalid. Each GS code is considered a valid SOR code or EOR code and will be bypassed.

MTDI Editing Restrictions

Following are the restrictions that apply when editing MTDI records:

- All canceled records are bypassed; they are not passed to any exit routines or written on any data sets. The level status is set to 0.

- All input records less than three bytes in length (SOR location, one data byte, and EOR location) are treated as canceled records. The remaining portion of a record that was longer than the user-specified maximum record size can result in an input record of this size.

- Data duplication is accomplished by replacing the DUP (duplication) code with the character from the corresponding location of the previous record.

- The record used for data duplication is the record returned from any user exits.

- GS codes will not affect the level status or duplication of following records.

- Data duplication does not occur for any of the following conditions:

  1. The DUP code is encountered in the first record of a cartridge.

  2. The DUP code is encountered in a record immediately following a canceled record. A canceled record is one that contains a cancel code in the SOR location or an input record of less than three bytes as described above.

  3. The DUP code is encountered in a position that would cause duplication of a position beyond the last data byte of the previous record.

  4. The DUP code is encountered in a position that would cause duplication of an error-replace character.

  In each case, the DUP code is replaced with the user specified error-replace character, and a field error is indicated.

- Left-zero justification does not occur; the left-zero fill code (LZ) is replaced with the user-specified error-replace character and a field error is indicated for either of the following conditions:

  1. The left-zero fill code (LZ) is encountered without first having encountered its corresponding left-zero start code (LZS).

  2. The user-specified maximum record size is exceeded before encountering the valid end of a left-zero field.

If MTDI is edited, an EDW which is four bytes long is appended to the front of each error record describing the error condition. For further definition of the EDW, see "Error Description Word (EDW)" on page 274. If the SYSUT3 DD statement specified variable length records, an RDW which is four bytes long is also appended to the front of the record. For further description of the RDW, see Supervisor Services and Macro Instructions.

The user-supplied routines specified in ERROR and OUTREC can be used to examine and modify any byte in the record or EDW. The record length can be changed, subject to the following restrictions:

- A work area used to construct the records is allocated by the program equal in size to the largest of (1) MAXLN, (2) LRECL on SYSUT2, or (3) LRECL on SYSUT3.
- The record length must not be increased beyond this size. Overlaying of other work areas may then occur, causing unpredictable results.

The new record length must be placed in the location pointed to by the second parameter word as received at entry to the routine. This length must include the EDW and RDW (if applicable). It is not necessary to modify the RDW because it is re-created if the record is to be written by IEBTCRIN. However, if the user does his own output from this routine, he must ensure that the RDW is correct for the record.

If IEBTCRIN is to write the record, the length of the output record depends on the RECFM specification, as follows:

- Fixed and variable records may have a maximum length equal to LRECL. Records larger than this are truncated.
- Undefined records may have a maximum length equal to BLKSIZE. Records larger than this are truncated.

These record lengths include the EDW and RDW, where applicable.

The record length returned from the error exit is used to establish the location of the last data byte in the record. The location is used to control data duplication in the following record. However, it is not used for checking the record length of subsequent records.

Modifications to the EDW, record, or record length may affect the editing of subsequent records. If the input is not edited, the user can examine and modify any byte in the record. The record length can also be changed, subject to the MTDI-editing restrictions.

SPECIAL CODES

Figure 87 on page 270 shows the hexadecimal characters representing special purpose codes that must not be used as replacement bytes.

MTDI Codes

X'00'	(LZ)	X'1E'	(VOK)	X'74'	(P4)
X'11'	(DUP)	X'3C'	(RM)	X'75'	(P5)
X'12'	(LZS)	X'71'	(P1)	X'76'	(P6)
X'18'	(CAN)	X'72'	(P2)	X'77'	(P7)
X'1D'	(GS)	X'73'	(P3)	X'78'	(P8)

MTST Codes

X'10'	(cr)	X'14'	(CR)	X'51'	(as)
X'11'	(sw)	X'15'	(SW)	X'55'	(AS)
X'13'	(fd)	X'17'	(FD)	X'80'	(src)
				X'81'	through X'FF'

Figure 87. Special Purpose Codes

The special purpose codes listed in Figure 87 are used by IEBTCRIN when constructing records. Use of these codes causes a message to be issued and the utility to be terminated.

Figure 88 on page 271 shows the values that can be chosen to replace error bytes for MTDI input.

Figure 89 on page 272 shows the values that can be chosen to replace error bytes for MTST input.

Figure 90 on page 273 shows MTST codes after they have been translated by IEBTCRIN when TRANS=STDLC is specified.

Bit Positions 4, 5, 6, 7	Second Hexadecimal Digit	00				01				10				11			
		00	01	10	11	00	01	10	11	00	01	10	11	00	01	10	11
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	0	LZ				SP	&								0	082	0
0001	1		DUP					/	P1					A	J		1
0010	2		LZS						P2					B	K	S	2
0011	3								P3					C	L	T	3
0100	4								P4					D	M	U	4
0101	5								P5					E	N	V	5
0110	6								P6					F	O	W	6
0111	7								P7					G	P	X	7
1000	8		CAN						P8					H	Q	Y	8
1001	9		ED											I	R	Z	9
1010	A					¢	!		:								
1011	B					.	$	,	#								
1100	C				RM	<	*	%	@								
1101	D		GS			(	)	-	/								
1110	E		VOK			+	;	>	=								
1111	F					|	¬	?	"								

Bit Positions 0, 1

Bit Positions 2, 3

First Hexadecimal Digit

Special Control:

LZ = Left zero fill
DUP = Duplicate
LZS = Left zero start
ED = End data
GS = Group Separator

Start of Record (SOR):

P1 = Program level 1
P2 = Program level 2
P3 = Program level 3
P4 = Program level 4
P5 = Program level 5
P6 = Program level 6
P7 = Program level 7
P8 = Program level 8
CAN = Cancel

End of Record (EOR):

RM = Record mark
VOK = Verify OK

This figure represents the character set and control codes as read from an MTDI created cartridge.

Figure 88. MTDI Codes from TCR

Bit Positions 4, 5, 6, 7	Second Hexadecimal Digit	00				01				10				11			
		00	01	10	11	00	01	10	11	00	01	10	11	00	01	10	11
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	0	z	cr	5	0	l	tab	'	s	src							
0001	1	2	sw	6	9	.	as	i	w								
0010	2	t		e	h	j	sp	p	y								
0011	3	n	fd	k	b	=		q	-								
0100	4	Z	CR	%	)	°	TAB	"	S	SRC							
0101	5	@	SW	¢	(	•	AS	I	W								
0110	6	T		E	H	J	SP	P	Y								
0111	7	N	FD	K	B	+		Q	-								
1000	8	1		7	4	m	bsp	r	o								
1001	9	3	st	8		v		a									
1010	A	x		d	l	g		:	/								
1011	B	u		c		f	stx	,									
1100	C	±		&	$	M	BSP	R	O								
1101	D	#	ST	*		V		A									
1110	E	X		D	L	G		:	?								
1111	F	U		C		F	STX	,									

Bit Positions 0, 1

Bit Positions 2, 3

First Hexadecimal Digit

cr and CR = Carrier return code
sw and SW = Switch code
fd and FD = Feed code
st and ST = stop code
tab and TAB = Tab code
as and AS = Automatic search
sp and SP = Space
bsp and BSP = Backspace
stx and STX = Stop transfer
src and SRC = Search

This figure represents the character set and control codes as read from an MTST created cartridge.

Figure 89. MTST Codes from TCR

Bit Positions 4, 5, 6, 7	Second Hexadecimal Digit	00				01				10				11				Bit Positions 0,1
		00	01	10	11	00	01	10	11	00	01	10	11	00	01	10	11	Bit Positions 2, 3
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	First Hexadecimal Digit
0000	0					SP	&										0	
0001	1							/		a	j	°		A	J		1	
0010	2			STX						b	k	s		B	K	S	2	
0011	3									c	l	t		C	L	T	3	
0100	4									d	m	u		D	M	U	4	
0101	5	TAB								e	n	v		E	N	V	5	
0110	6		BSP							f	o	w		F	O	W	6	
0111	7									g	p	x		G	P	X	7	
1000	8									h	q	y		H	Q	Y	8	
1001	9									i	r	z		I	R	Z	9	
1010	A					¢	!		:									
1011	B						$	.	#									
1100	C						*	%	@									
1101	D	CR				(	)	_	,									
1110	E		SRC			+	;		=		±							
1111	F							?	"									

TAB = Tab code
CR = Carrier return
BSP = Backspace
SRC = Search
STX = Stop transfer
SP = Space

Note: The STDUC option permits translating both lowercase and uppercase alphabetic characters to uppercase.

Figure 90. MTST Codes after Translation by IEBTCRIN with TRANS=STDLC

END-OF-CARTRIDGE

Unique codes, written by the MTST or the MTDI device, signal the program when all data on a cartridge has been read. For MTST cartridges, this end-of-cartridge code is a lowercase stop code (st) or an uppercase stop code (ST). For MTDI cartridges, the end-of-cartridge code is the end-data code (ED).

IEBTCRIN terminates input from a cartridge upon encountering the end-of-cartridge code and rewinds the cartridge. IEBTCRIN continues to process cartridges until end-of-file is encountered.

End-of-file is signaled following a rewind operation when there are no more cartridges in the feed hopper, the END OF FILE button is pressed, and end-of-cartridge for the last cartridge is recognized. An end-of-file indication will be passed to the OUTREC and/or ERROR exits if specified by setting register 1 equal to 0.

A stop code, whether uppercase (ST) or lowercase (st), indicates that all data on a cartridge has been read. Therefore, when an MTST cartridge to be processed by IEBTCRIN is created, the user must not use a stop code for any purpose other than signaling end-of-data on the cartridge. Stop codes within meaningful data cause any subsequent data on the cartridge to be lost because the cartridge is rewound and unloaded when a stop code is encountered.

ERROR RECORDS

If a record is found to be in error, the record is passed to the user error exit routine if one is specified. If an error exit is not specified, the action to be taken is determined by the option specified in a utility control statement.

When either MTST input or MTDI input without editing is specified, the only error that can be recognized is a record containing one or more permanent data checks. The data check bytes are replaced as described in a utility control statement. The record is considered an error record, but because a data check is the only error that can occur, no EDW is appended to the error record.

ERROR DESCRIPTION WORD (EDW)

The Error Description Word (EDW) consists of four bytes that are appended to the start of an error record.

The error description word is in EBCDIC format; for example, a 2 is represented as X'F2' and a C is represented as X'C3'. The information provided in each of the four bytes of the EDW is discussed in the following table.

Byte	Indicator Meaning
Level Status (Byte 0)	Identifies error records that result from interrecord dependency that cannot be identified in the type status byte. **Value Meaning** **0** Indicates any error record that will not cause questionable data in the following records. A type status other than zero accompanies this byte. **1** Indicates any error record that may cause questionable data in the following records, and for which the level status of the previous record was 0. **2** Indicates any error that contains questionable data because the error level of the preceding record was 1 or 2, or for any error record that may cause questionable data in the following records and for which the level status of the previous record was 1 or 2. A level status of 1 or 2 is presented with error records resulting from the following: • The start-of-record (SOR) location has a character defined as an error. • The record contains two or more data check bytes side by side. These may have been an SOR and EOR (end-of-record). • The record is longer than the user-specified maximum length record. • The length of the record is not equal to the length of the first valid record of the same program level encountered on this cartridge. For this purpose, a valid record is one that contains no errors as identified in the type status, with the possible exception of being shorter than the user-specified minimum length. • The record has a data-duplication dependency on a previous record with one of the above errors. The level status is set to 0 when IEBTCRIN encounters: (1) a record without one of the previous errors, (2) a canceled record, or (3) the first record of a cartridge.

Byte	Indicator Meaning
Type Status (Byte 1)	Identifies records in error because of SOR, EOR, length, field, or data check error conditions.
	Value **Meaning**
	0 Indicates any record that contains none of the following identifiable errors, but contains questionable data due to a level status other than zero. (See Level Status above.)
	1 Indicates any record that has: (1) an SOR character of other than P1 through P8 or a GS code, (2) an EOR character of other than a VOK code for records when the user specified a record verification check, or (3) an EOR character of other record-verification check.
	2 Indicates any record that has an incorrect length because it is: (1) longer than the user-specified maximum, (2) shorter than the user-specified minimum, or (3) not encountered on this cartridge.
	4 Indicates any record that has a field error. A field error occurs when duplication or left-zero justification functions did not occur in a field because of an error condition. See "MTDI Editing Criteria" below.
	8 Indicates any record that has a permanent data check error.
	The type-status indicator can also have values of 3, 5, 6, 7, 9, A, B, C, D, E, and F. These values indicate a combination of SOR, EOR, length, field, and data check errors. For example, a value of A indicates a record with a data check error (8), as well as, an incorrect length (2).
Start-of-Record (Byte 2)	Indicates the start-of-record (SOR) character associated with this record. The SOR character can be 1 through 8, where 1 indicates P1, 2 indicates P2, etc., or E, which indicates the SOR character is in error.
End-of-Record (Byte 3)	Indicates the end-of-record (EOR) character associates with this record. The EOR character can be: U (unverified record); V (verified record); or E (EOR character is in error).

SAMPLE ERROR RECORDS

Figure 91 shows a stream of data bytes read sequentially from the tape cartridge reader.

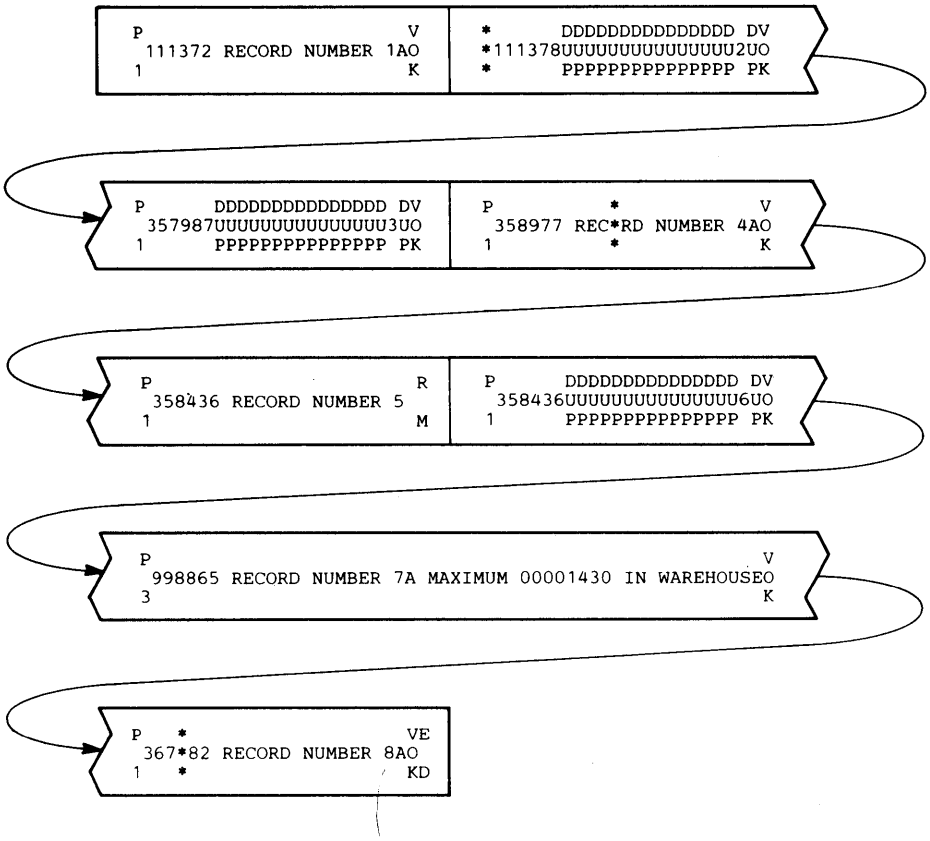


Figure 91. Tape Cartridge Reader Data Stream

Figure 92 on page 278 shows the records constructed by IEBTCRIN from the input records shown in Figure 91. These records show some of the errors that can occur during processing and their effect on the Error Description Word. The following parameters were specified for these records:

```
                                                                  72
TCRGEN      TYPE=MTDI,EDIT=EDITR,VERCHK=VOKCHK,                    C
            MAXLN=50,REPLACE=X'5B'
```

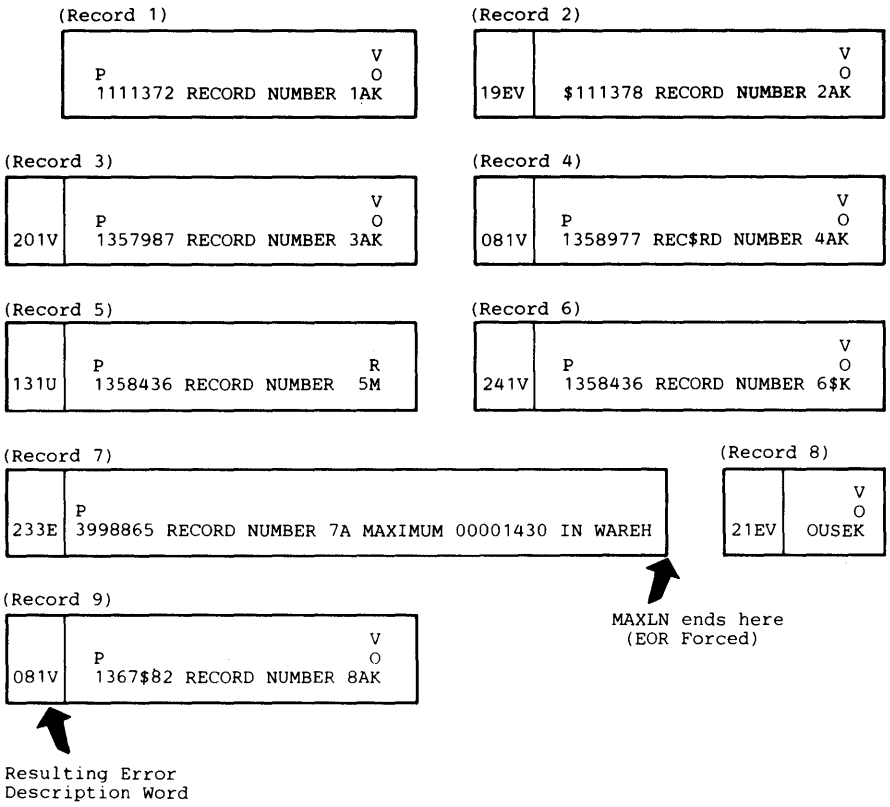


Figure 92. Record Construction

IEBTCRIN classifies records 2 through 9 in Figure 92 as error records. The records are classified as follows:

- Record 1 is a valid record. It contains a program-level 1 code, and thus establishes the valid length for all program-level 1 records in this cartridge to be 25 bytes.
- Record 2 has a data check in the SOR location. Level status is set to 1 because the SOR location might have contained a cancel code that would cause any data duplicated on the following record to be questionable. The type status (9) indicates the record has an incorrect SOR/EOR character (1) and a data check error (8).
- Record 3 contains no identifiable error, but contains questionable data because it requires duplication from the previous record, which had a level status of 1.
- Record 4 has a data check. Because it contained no DUP codes, the level status is set to 0.
- Record 5 is shorter than the first program-level 1 record on this cartridge (length error). This record also contains an RM code rather than a VOK code in the EOR location (VOKCHK was specified on the TCRGEN statement. Because IEBTCRIN cannot determine why the record is short, all data duplicated from this record is questionable; the level status is set to 1. The type status is set to 3 indicating an SOR/EOR error (1) and length error (2).

- Record 6 contains a DUP code that is beyond the last position of the preceding record.
- The seventh input record is longer than the maximum user-specified record length. Note that it is passed as two records. The first record (record 7) indicates an EOR error and a length error; the second (record 8) indicates an SOR error. Because record 7 is an error record, its length (50 bytes) is not established as the valid length for all program-level 3 records on this cartridge.
- Record 9 has a data check. Because it contained no DUP codes, the level status is set to 0.

INPUT AND OUTPUT

IEBTCRIN uses the following input:

- An input data set, which contains data on tape cartridges to be read from the Tape Cartridge Reader (TCR). The input data set was created on either MTST or MTDI.
- A control data set, which contains utility control statements that are used to control the functions of IEBTCRIN.

IEBTCRIN produces the following output:

- An output data set, which contains the sequential output produced by the utility as a result of processing the cartridge input according to the utility control statements.
- An error output data set, which contains records that do not conform to the specifications for a valid record.
- A message data set, which contains diagnostic messages.

RETURN CODES

IEBTCRIN returns a code in register 15 to indicate the results of program execution. The return codes and their meanings are listed below.

Codes	Meaning
00 (00 hex)	Normal termination.
04 (04)	Warning message issued; execution permitted. Conditions leading to issuance of this code are: • SYSPRINT, SYSIN, SYSUT2, or SYSUT3 DD statements missing and • DCB parameters missing SYSUT2 or SYSUT3 DD statements.
12 (0C)	Diagnostic error message issued; execution terminated. Conditions leading to issuance of this code are: • SYSUT1 DD statement missing, • Conflicting DCB parameters in DD statements, and • Invalid or conflicting utility control statements.
16 (10)	Terminal error message issued; execution terminated. Conditions leading to issuance of this code are: • Permanent input/output errors (not including data checks on the TCR), • Unsuccessful opening of data sets, • Requests for termination by user exit routine, • Insufficient storage available for execution, and • User exit routine not found.

Figure 93. IEBTCRIN Return Codes

CONTROL

IEBTCRIN is controlled by job control statements and utility control statements. The job control statements are required to execute or invoke IEBTCRIN and to define the data sets that are used and produced by the program. The utility control statements are used to indicate the source of the input data cartridges (MTST or MTDI) and to specify the type of processing to be done.

JOB CONTROL STATEMENTS

Figure 94 on page 281 shows the job control statements for IEBTCRIN.

Statement	Use
JOB	Initiates the job.
EXEC	Specifies the program name (PGM=IEBTCRIN) or, if the job control statements reside in a procedure library, the procedure name.
SYSPRINT DD	Defines a sequential message data set, which can be written to any QSAM-supported output device.
SYSUT1 DD	Defines the input data set.
SYSUT2 DD	Defines a sequential output data set for valid records.
SYSUT3 DD	Defines a sequential output data set for error records.
SYSIN DD	Defines the control data set. The control data set normally resides in the input stream; however, it can be defined as a sequential data set or as a member of a partitioned data set. If this statement is not included, all utility control statement defaults are assumed and a message is issued to SYSPRINT. If DUMMY is specified, all utility control statement defaults are assumed.

Figure 94. IEBTCRIN Job Control Statements

SYSPRINT DD Statement

If the SYSPRINT DD statement is missing, a message is written on the operator console and processing continues.

If some parameters are specified but others are omitted, IEBTCRIN attempts to set defaults for the missing parameters that are consistent with those supplied. For example, if RECFM=VBA is specified, IEBTCRIN assumes BLKSIZE=129 and LRECL=125. If LRECL, BLKSIZE, and RECFM are not specified, the defaults are LRECL=121, BLKSIZE=121, and RECFM=FBA.

Because IEBTCRIN always constructs the SYSPRINT records with USASI (type A) control characters, type A control characters should be indicated when RECFM is specified.

The DCB parameters defining the SYSPRINT data set can be supplied from any valid source (for example, DD statements).

If a parameter that is not consistent with the other parameters is specified on SYSPRINT DD, a message is issued and processing is ended.

If a permanent input/output error occurs on SYSPRINT, both the failing message and a SYNADAF message indicating the error are written on the programmer's console and processing is terminated.

SYSUT1 DD Statement

The SYSUT1 DD statement is required for each use of IEBTCRIN.

For the SYSUT1 DD statement, only the UNIT keyword is required. The value specified in UNIT=xxxx can be '2495', the device address, or any other name that was generated in the system as a unit device name. The VOLUME=SER=keyword may be specified to identify the tape cartridges to be mounted. The volume serial

number must be an externally recognizable name associated with the cartridges to be processed. A message is issued to the operator instructing that the cartridges identified by that name be mounted. If VOLUME is not specified, the name TCRINP is assumed and used in the mount message. The BUFL DCB parameter can be specified to indicate the size of input buffers; if BUFL is not specified, a value of 2000 is assumed.

If a permanent error occurs on SYSUT1 (not including a data check), a message is issued on SYSPRINT and the program is terminated.

SYSUT2 and SYSUT3 DD Statements

The SYSUT2 DD and SYSUT3 DD statements must identify sequential data sets; the data sets can have fixed, variable, variable spanned, or undefined records. These data sets can be written on any QSAM-supported device.

Fixed and variable records on the SYSUT2 or SYSUT3 data set can be blocked through the specification of the BLKSIZE and RECFM DCB parameters.

SYSUT2 DD and SYSUT3 DD statements may be omitted or specified as DUMMY for other than sequential data sets. A message is issued on SYSPRINT and processing continues.

The DCB parameters defining the SYSUT2 and SYSUT3 data sets can be supplied from any valid source (for example, DD statements or a data set label). Because the output (SYSUT2 and/or SYSUT3) data sets are not opened until the first record is ready for output (after any OUTREC and/or ERROR exits), DCB parameters to be supplied from an existing data set label are not available for records constructed before the data set is opened. Therefore, the DCB parameters should always be provided in the DD statement even though they may already exist in the label. Otherwise, defaults are used to construct records until the data set is opened.

If editing of MTDI input is specified on the utility control statements, the SYSUT3 LRECL parameter should be four bytes greater than the SYSUT2 LRECL parameter to include a four bytes long Error Description Word appended to the front of the record by IEBTCRIN. (See "Error Description Word (EDW)" on page 274.) For variable records on either SYSUT2 or SYSUT3, the LRECL and BLKSIZE DCB parameters must be large enough to include the four bytes long record descriptor word.

If inconsistent parameters are specified on SYSUT2 DD or SYSUT3 DD, a message is issued and processing is ended.

If a permanent error occurs on SYSUT2 or SYSUT3, a message is issued on SYSPRINT and the program is terminated.

SYSIN DD Statement

The DCB parameters defining the SYSIN data set can be supplied from any valid source (for example, DD statements or a data set label). Because the output (SYSUT2 and/or SYSUT3) data sets are not opened until the first record is ready for output (after any OUTREC and/or ERROR exits), DCB parameters to be supplied from an existing data set label are not available for records constructed before the data set is opened. Therefore, the DCB parameters should always be provided in the DD statement even though they may already exist in the label. Otherwise, defaults are used to construct records until the data set is opened.

If a permanent error occurs on SYSIN, a message is issued on SYSPRINT and the program is terminated.

UTILITY CONTROL STATEMENTS

Figure 95 shows the utility control statements for IEBTCRIN.

Statement	Use
TCRGEN	Specifies whether MTDI or MTST input is to be processed and the type of processing to be performed.
EXITS	Specifies any exit routines provided by the user.

Figure 95. IEBTCRIN Utility Control Statements

Continuation requirements for utility control statements are described in "Continuing Utility Control Statements" on page 5.

If these statements contain errors or inconsistencies, the program is terminated and the appropriate diagnostics are sent to the message data set. If TCRGEN is not specified, standard defaults are used.

TCRGEN Statement

The TCRGEN statement is used to indicate the device (MTDI or MTST) on which the input data was created and the type of processing to be performed on the input data.

The format of the TCRGEN statement is:

[label]	TCRGEN	[TYPE=MTDI|MTST] [,TRANS=STDUC|STDLC|name|NOTRAN] [,EDIT=EDITD|EDITR|NOEDIT] [,VERCHK=NOCHK|VOKCHK] [,MINLN=n] [,MAXLN=n] [,REPLACE=X'xx'] [,ERROPT=NORMAL|NOERR]

EXITS Statement

The EXITS statement is used to identify user-supplied exit routines, which must exist in either the user job library or the link library.

Upon entry, a parameter list is supplied to the exit routine. Upon returning from the exit routine, the user must provide an acceptable return code. See Appendix A, "Exit Routine Linkage" on page 438.

The format of the EXITS statement is:

[label]	**EXITS**	**[ERROR=**routinename**]** **[,OUTREC=**routinename**]** **[,OUTHDR2=**routinename**]** **[,OUTHDR3=**routinename**]** **[,OUTTLR2=**routinename**]** **[,OUTTLR3=**routinename**]**

Parameters	Applicable Control Statements	Description of Parameters
EDIT	TCRGEN	**EDIT=EDITD|EDITR|NOEDIT** specifies the type of processing to be performed on MTDI input. These values can be coded: **EDITD** specifies that the input is to be edited and that SOR and EOR codes are to be deleted and not included as part of the output record. This is the default. **EDITR** specifies that the input is to be edited and SOR and EOR codes are to be kept as part of the output record. **NOEDIT** specifies that no editing is to be performed. Data, including any group separator (GS) codes, is passed exactly as read from the cartridge. If EDITD or EDITR is specified, the edit consists of the following functions: • Records are extracted one at a time from the input buffers by scanning for the record-delimiting codes (SOR and EOR). • DUP codes are replaced with the character from the corresponding location in the preceding record. • Left-zero fields are right aligned and leading zeros are inserted where necessary. • Left-zero start codes are deleted from the records. • Group separator codes and records that start with cancel record codes are bypassed.

Parameters	Applicable Control Statements	Description of Parameters
ERROPT	TCRGEN	**ERROPT=NORMAL|NOERR** specifies the disposition of all error records. ERROPT is ignored if a user error routine is specified in the EXITS statement. These values can be coded: **NORMAL** specifies that all error records are to be placed in the error data set (SYSUT3). **NOERR** specifies that all records (including error records) are placed in the normal output data set (SYSUT2). No records are placed in the error data set (SYSUT3). This is the default.
ERROR	EXITS	**ERROR=routinename** specifies the name of the routine that receives control before an error record is passed to the error output data set (SYSUT3). This exit routine can be used to analyze and, if possible, correct the error record. This parameter nullifies any ERROPT value.

Parameters	Applicable Control Statements	Description of Parameters
MAXLN	TCRGEN	**MAXLN=n** specifies the number of bytes, n, plus four for the record descriptor word when variable records are specified, to be contained in all but the last record passed to the output routine when editing is not performed. IEBTCRIN does not indicate the end of data from one cartridge and the beginning of data from the next. Usually this transition from one cartridge to another occurs within an output record. The last record passed to the output routine contains only the number of bytes remaining (plus four if the record format is variable) and is the only record that can be shorter than the length specified by MAXLN. The size of the records actually written depends on the record length (LRECL) specified for the output data set. For MTDI input with editing specified, MAXLN is used to specify in bytes the length of the longest valid record after editing. If the program encounters a record in which a valid end-of-record cannot be determined within this length, an end-of-record condition is forced and the record is considered an error record. The values that can be specified for MAXLN are: • For MTST processing or MTDI processing without editing, MAXLN should equal the number of bytes to be passed as a record. • For MTDI processing when EDIT=EDITD, MAXLN should equal the number of bytes in the longest valid record after editing, excluding SOR and EOR codes. • For MTDI processing when EDIT=EDITR, MAXLN should equal the number of bytes in the longest valid record after editing, including SOR and EOR codes. **Note:** The values for MAXLN should not include the 4-byte-long record descriptor word added to a variable length record. **Default:** 120 bytes

Parameters	Applicable Control Statements	Description of Parameters
MINLN	TCRGEN	**MINLN=**n specifies in bytes the length, n, of the shortest valid edited record. This parameter is valid only when TYPE=MTDI and either EDIT=EDITD or EDIT=EDITR are specified. If IEBTCRIN encounters a record shorter than this specified length, the record is considered an error record. The values that can be specified for MINLN are: • For MTST processing or MTDI processing without editing, MINLN is not specified. • For MTDI processing when EDIT=EDITD, MINLN should equal the number of bytes in the shortest valid record after editing, excluding SOR and EOR codes. • For MTDI processing when EDIT=EDITR, MINLN should equal the number of bytes in the shortest valid record after editing, including SOR and EOR codes. **Note:** The values for MINLN should not include the four bytes long record descriptor word added to a variable length record. **Default:** No minimum length checking is performed.
OUTREC	EXITS	**OUTREC=**routinename specifies the name of the routine that receives control before the record is passed to the normal output data set (SYSUT2). In this exit routine, the user can process the record and perform his own output if output other than the SYSUT2 data set is desired. Any modification of an edited MTDI record may affect the editing of following records. The record returned from this exit is used to accomplish data duplication in the record that follows. If the SYSUT2 data set has specified variable length records, an RDW which is four bytes long is appended to the front of the record.
OUTHDR2	EXITS	**OUTHDR2=**routinename specifies the name of the routine that receives control during the opening of the SYSUT2 data set; this exit routine can be used to create user output header labels for the normal output data set (SYSUT2).
OUTHDR3	EXITS	**OUTHDR3=**routinename specifies the name of the routine that receives control during the opening of the SYSUT3 data set; this exit routine can be used to create user output header labels for the error data set (SYSUT2).

Parameters	Applicable Control Statements	Description of Parameters
OUTTLR2	EXITS	**OUTTLR2=**routinename specifies the name of the routine that receives control during the closing of the SYSUT2 data set; this exit routine can be used to create user output trailer labels for the normal output data set (SYSUT2).
OUTTLR3	EXITS	**OUTTLR3=**routinename specifies the name of the routine that receives control during the closing of the SYSUT3 data set; this exit routine can be used to create user output trailer labels for the error data set (SYSUT3).
REPLACE	TCRGEN	**REPLACE=X'**xx**'** specifies the hexadecimal representation of the character to be used by IEBTCRIN to replace error bytes. REPLACE allows the user to identify and possibly correct error bytes on the error exit routine or in subsequent processing. The specified REPLACE character should be one that does not normally appear in the data. To replace error bytes on MTDI data, select a value for xx from Figure 88 on page 271 . To replace error bytes on MTST data, select a value for xx from Figure 89 on page 272 . The replacement of error bytes is accomplished before any specified MTST translation. **Default:** X'19', end-of-data

Parameters	Applicable Control Statements	Description of Parameters
TRANS	TCRGEN	**TRANS=STDUC|STDLC|name|NOTRAN** specifies the type of processing to be performed on MTST input. These values can be coded: **STDUC** specifies that the MTST code is to be translated to standard EBCDIC; alphabetic characters are translated to uppercase. This is the default. **STDLC** specifies that the MTST code is to be translated to standard EBCDIC; alphabetic characters are not translated to uppercase. name specifies a user translate table to be used by IEBTCRIN. The translate table must exist as a load module named in a user job library or the link library. This load module must consist of a translate table which begins at the entry point and conforms to the specifications for the translate instruction (TR) found in IBM System/370 Principles of Operation. **NOTRAN** specifies that no translation and no special processing are to be performed. Data is passed exactly as read from the cartridge. If STDUC, STDLC, or name is specified, certain of the MTST codes are processed in a special way before translation. Feed codes (FD), switch codes (SW), and autosearch codes (AS), both uppercase and lowercase, are deleted from the data. Each 61-character reference code is reduced to a single search code (SRC). See "Special Codes" on page 269 for an explanation of these codes.
TYPE	TCRGEN	**TYPE=MTDI|MTST** specifies the device on which the magnetic tape cartridge(s) was written. These values are coded: **MTDI** specifies that the input was created on a Magnetic Data Inscriber. This is the default. **MTST** specifies that the input was created on a Magnetic Tape Selectric typewriter.

Parameters	Applicable Control Statements	Description of Parameters
VERCHK	TCRGEN	**VERCHK=NOCHK|VOCHK** specifies whether a record-verification check is to be made on MTDI input that is to be edited. This parameter is valid only when TYPE=MTDI and either EDIT=EDITD or EDIT=EDITR are specified. These values can be coded: **NOCHK** specifies that no record-verification check is to be made. Either a record mark (RM) or a verify OK (VOK) code is considered a valid end-of-record code. This is the default. **VOKCHK** specifies that a record-verification check is to be made. A record that does not contain a verify OK code is to be considered an error record.

IEBTCRIN EXAMPLES

The following examples illustrate some of the uses of IEBTCRIN. Figure 96 can be used as a quick reference guide to IEBTCRIN examples. The numbers in the "Example" column point to examples that follow.

Operation	Data Set Organization	Device	Comments	Example
Edit MDTI input	Sequential	Disk and 9-track Tape	Fixed blocked output. Error exit routine specified	1
Invoke IEBTCRIN with LINK macro instruction	—	—	Assembler language interface instructions	2

Figure 96. IEBTCRIN Example Directory

Examples that use **disk** or **tape** in place of actual device numbers must be changed before use. See "DASD and Tape Device Support" on page 3 for valid device number notation.

IEBTCRIN EXAMPLE 1

In this example, input from a tape cartridge is to be edited with normal records written to a disk volume and error records written to a tape volume.

```
//JOBNAME  JOB    O,SMITH,MSGLEVEL=1
//STPNAME  EXEC   PGM=IEBTCRIN
//SYSPRINT DD     SYSOUT=A
//SYSUT1   DD     UNIT=TCR,VOLUME=SER=MYTAPE,DCB=(BUFL=3000)
//SYSUT2   DD     DSNAME=GOODSET,DISP=(NEW,CATLG),UNIT=disk
//                VOLUME=SER=111222,SPACE=(TRK(10,10)),
//                DCB=(LRECL=100,BLKSIZE=1000,RECFM=FB)
//SYSUT3   DD     DSNAME=ERRSET,UNIT=tape,VOLUME=SER=000001,
//                DISP=(NEW,KEEP),DCB=(BLKSIZE=104,RECFM=U)
//SYSIN    DD     *
       TCRGEN TYPE=MTDI,EDIT=EDITD,MAXLN=100,REPLACE=X'5B'
       EXITS  ERROR=MYERR
/*
```

The control statements are discussed below:

- SYSUT1 DD defines the input tape cartridge data set. A console message instructs the operator to mount a set of cartridges named MYTAPE. The two input buffers are each 3000 bytes long (BUFL). The UNIT parameter assumes that TCR has been system generated as an esoteric name for the Tape Cartridge Reader.
- SYSUT2 DD defines a sequential data set for the normal output records. The data will be written to a disk volume.
- SYSUT3 DD defines a sequential data set for the error records. The records are undefined with a maximum block size of 104 bytes, including a 4-byte error description word.
- SYSIN DD defines the control data set, which follows in the input stream.
- TCRGEN indicates MTDI input. The input is to be edited with SOR and EOR codes deleted, the maximum valid record length is to be 100 bytes, and the replace character is a hexadecimal "5B." VERCHK is defaulted to NOCHK. Minimum record length checking is not requested.
- EXITS indicates that a user has provided an exit routine to handle error records. Because no job library has been specified, the exit routine (MYERR) must reside in the link library.

IEBTCRIN EXAMPLE 2

In this example, IEBTCRIN is invoked via the LINK macro instruction in an Assembler language program. An alternate name has been assigned to each of the DD statements used by IEBTCRIN. The job control for this step must include DD statements with the alternate DD names.

```
         LINK  EP=IEBTCRIN,PARAM=(OPTLIST,DDNAME),VL=1
         CNOP  2,4    (OPTLIST must be on halfword boundary)
OPTLIST  DC    H'0'   (Length must be zero for IEBTCRIN)
         CNOP  2,4    (DDNAME list must be on halfword boundary)
DDNAME   DC    H'82'  (Length of DDNAME list)
         DC    8F'0'
         DC    C'NEWIN    ' (Alternate DDNAME for SYSIN)
         DC    C'NEWPRINT' (Alternate DDNAME for SYSPRINT)
         DC    2F'0'
         DC    C'NEWUT1   ' (Alternate DDNAME for SYSUT1)
         DC    C'NEWUT2   ' (Alternate DDNAME for SYSUT2)
         DC    C'NEWUT3   ' (Alternate DDNAME for SYSUT3)
```

IEBUPDTE PROGRAM

IEBUPDTE is a data set utility used to incorporate IBM and user-generated source language modifications into sequential or partitioned data sets. Exits are provided for user routines that process user header and trailer labels.

IEBUPDTE can be used to:

- Create and update data set libraries
- Modify existing partitioned members or sequential data sets
- Change the organization of a data set from sequential to partitioned or vice versa

CREATING AND UPDATING DATA SET LIBRARIES

IEBUPDTE can be used to create a library of partitioned members consisting of (at the most) 80-byte logical records. In addition, members can be added directly to an existing library, provided that the original space allocations are sufficient to incorporate the new members. In this manner, a cataloged procedure can be placed in a procedure library, or a set of job or utility control statements can be placed as a member in a partitioned library.

MODIFYING AN EXISTING DATA SET

IEBUPDTE can be used to modify an existing partitioned or sequential data set. Logical records can be replaced, deleted, renumbered, or added to the member or data set.

A sequential data set residing on a tape volume can be used to create a new master (that is, a modified copy) of the data set. A sequential data set residing on a direct access device can be modified either by creating a new master or by modifying the data set directly on the volume on which it resides.

A partitioned data set can be modified either by creating a new master or by modifying the data set directly on the volume on which it resides.

CHANGING DATA SET ORGANIZATION

IEBUPDTE can be used to change the organization of a data set from sequential to partitioned, or to change a single member of a partitioned data set to a sequential data set. If only a member is changed, the remainder of the original data set remains unchanged. In addition, logical records can be replaced, deleted, renumbered, or added to the member or data set.

INPUT AND OUTPUT

IEBUPDTE uses the following input:

- An input data set (also called the old master data set), which is modified or used as source data for a new master. The input data set is either a sequential data set or a member of a partitioned data set.
- A control data set, which contains utility control statements and, if applicable, input data. The data set is required for each use of IEBUPDTE.

IEBUPDTE produces the following output:

- An output data set, which is the result of the IEBUPDTE operation. The data set can be either sequential or partitioned. It can be either a new data set (that is, created during the present job step) or an existing data set, modified during the present job step.
- A message data set, which contains the utility program identification, control statements used in the job step, modification made to the input data set, and diagnostic messages, if applicable. The message data set is sequential.

RETURN CODES

IEBUPDTE returns a code in register 15 to indicate the results of program execution. The return codes and their meanings are listed below.

Codes	Meaning
00 (00 hex)	Successful completion.
04 (04)	A control statement is coded incorrectly or used erroneously. If either the input or output is sequential, the job step is terminated. If both are partitioned, the program continues processing with the next function to be performed.
12 (0C)	An unrecoverable error exists. The job step is terminated.
16 (10)	A label processing code of 16 was received from a user's label processing routine. The job step is terminated.

Figure 97. IEBUPDTE Return Codes

CONTROL

IEBUPDTE is controlled by job control statements and utility control statements. The job control statements are required to execute or invoke IEBUPDTE and to define the data sets that are used and produced by the program. The utility control statements are used to control the functions of IEBUPDTE and, in certain cases, to supply new or replacement data.

JOB CONTROL STATEMENTS

Figure 98 on page 295 shows the job control statements for IEBUPDTE.

Statement	Use
JOB	Initiates the job.
EXEC	Specifies the program name (PGM=IEBUPDTE), or, if the job control statements reside in a procedure library, the procedure name. Additional information can be specified in the PARM parameter of the EXEC statement.
SYSPRINT DD	Defines a sequential message data set. The data set can be written to a system output device, a tape volume, or a direct access volume.
SYSUT1 DD	Defines the input (old master) data set. It can define a sequential data set on a card reader, a tape volume, or a direct access volume. Or, it can define a partitioned data set on a direct access volume.
SYSUT2 DD	Defines the output data set. It can define a sequential data set on a card punch, a printer, a tape volume, or a direct access device. It can define a partitioned data set on a direct access device.
SYSIN DD	Defines the control data set. The control data set normally resides in the input stream; however, it can be defined as a member of a partitioned data set.

Figure 98. Job Control Statements for IEBUPDTE

PARM Information on the EXEC Statement

Additional information can be coded in the PARM parameter of the EXEC statement, as follows:

```
EXEC¬PAR |={NEW|MOD}[,inhdr][,intlr]
```

Following are the PARM values:

- NEW, which specifies that the input consists solely of the control data set. An input data set does not exist and is not defined if NEW is specified.
- MOD, which specifies that the input consists of both the control data set and the input data set. If neither NEW nor MOD is coded, MOD is assumed.
- inhdr, which specifies the name of the routine that processes the user header label on the volume containing the control data set.
- intlr, which specifies the name of the routine that processes the user trailer label on the volume containing the control data set.

SYSPRINT DD Statement

The message data set has a logical record length of 121 bytes, and consists of fixed length, blocked or unblocked records with an American National Standards Institute (ANSI) control character in the first byte of each record. The input and

output data sets have a logical record length of 80 bytes or less, and consist of standard fixed-blocked (RECFM=FB) or unblocked records. The control data set contains 80-byte, blocked or unblocked records.

SYSUT1 DD Statement

If the SYSUT1 and SYSUT2 DD statements define the same sequential data set (BDAM only), only those operations that add data to the end of the existing data set can be made. In these cases:

1. The PARM parameter of the EXEC statement must imply or specify MOD. (See "PARM Information on the EXEC Statement" on page 295.)

2. The DISP parameter of the SYSUT1 DD statement must specify OLD.

If SYSUT1 and SYSUT2 define the same partitioned data set, new extents resulting from updates on SYSUT2 are not retrievable in SYSUT1.

The input and output data sets contain blocked or unblocked logical records with record lengths of up to 80 bytes. The input and output data sets may have different block sizes as long as they are multiples of the logical record length.

If an ADD operation is specified with PARM=NEW in the EXEC statement, the SYSUT1 DD statement need not be coded.

If the SYSUT1 DD statement defines a sequential data set on tape, the file sequence number of that data set must be included in the LABEL keyword (unless the data set is the first or only data set on the volume).

SYSUT2 DD Statement

Space must be allocated for an output data set (SYSUT2 DD statement) that is to reside on a direct access device, unless the data set is an existing data set.

The SYSUT2 DD statement must not specify a DUMMY data set.

When adding a member to an existing partitioned data set using an ADD function statement, any DCB parameters specified on the SYSUT1 and SYSUT2 DD statements (or the SYSUT2 DD statement if that is the only one specified) must be the same as the DCB parameters already existing for the data set.

If the SYSUT1 and SYSUT2 DD statements define the same sequential data set (BDAM only), only those operations that add data to the end of the existing data set can be made. In these cases:

1. The PARM parameter of the EXEC statement must imply or specify MOD. (See "PARM Information on the EXEC Statement" on page 295.)

2. The DISP parameter of the SYSUT2 DD statement must specify MOD.

If SYSUT1 and SYSUT2 define the same partitioned data set, new extents resulting from updates on SYSUT2 are not retrievable in SYSUT1.

The output data set can have a blocking factor that is different from the input data set; however, if insufficient space is allocated for reblocked records, the update request is terminated.

The input and output data sets contain blocked or unblocked logical records with record lengths of up to 80 bytes. The input and output data sets may have different block sizes as long as they are multiples of the logical record length.

If an UPDATE=INPLACE operation is specified, the SYSUT2 DD statement should not be coded.

If both the SYSUT1 and SYSUT2 DD statements specify standard user labels (SUL), IEBUPDTE copies user labels from SYSUT1 to SYSUT2.

If the SYSUT1 and SYSUT2 DD statements define the same partitioned data set, the old master data set can be updated without creating a new master data set; in this case, a copy of the updated member or members is written within the extent of the space originally allocated to the old master data set. Subsequent referrals to the updated member(s) will point to the newly written member(s). The member names themselves should not appear on the DD statements; they should be referred to only through IEBUPDTE control statements. The old directory entry for each member is not copied.

SYSIN DD Statement

The SYSIN DD statement is required for each use of IEBUPDTE.

UTILITY CONTROL STATEMENTS

Figure 99 shows the utility control statements used to control IEBUPDTE.

Statement	Use
Function	Initiates an IEBUPDTE operation (ADD, CHANGE, REPL, REPRO).
Detail	Used with the function statement for special applications.
Data	A logical record of data to be used as a new or replacement record in the output data set.
LABEL	Indicates that the following data statements are to be treated as user labels.
ALIAS	Assigns aliases.
ENDUP	Terminates IEBUPDTE.

Figure 99. IEBUPDTE Utility Control Statements

Note: Unlike other utility control statements, all IEBUPDTE utility control statements (including the NUMBER and DELETE detail statements, but not including data statements) must begin with a "./" (period, slash) in columns 1 and 2.

Continuation requirements for utility control statements are described in "Continuing Utility Control Statements" on page 5.

Function Statement

The function statement (ADD, CHANGE, REPL, or REPRO) is used to initiate an IEBUPDTE operation. At least one function statement must be provided for each member or data set to be processed.

A member or a data set can be added directly to an old master data set if the space originally allocated to the old master is sufficient to incorporate that new member or data set. ADD specifies that a member or a data set is added to an old master data set. If a member is added and the member name already exists in the old master data set, processing is terminated. If, however, PARM=NEW is specified on the EXEC statement, the member is replaced. For a sequential output master data set, PARM=NEW must always be specified on the EXEC statement. At least one blank must precede and follow ADD.

When a member replaces an identically named member on the old master data set or a member is changed and rewritten on the old master, the alias (if any) of the original member still refers to the original member. However, if an identical alias is specified for the newly written member, the original alias entry in the directory is changed to refer to the newly written member.

REPL specifies that a member of a data set is being entered in its entirety as a replacement for a sequential data set or for a member of the old master data set. The member name must already exist in the old master data set. At least one blank must precede and follow REPL. CHANGE specifies that modifications are to be made to an existing member or data set. Use of the CHANGE function statement without a NUMBER or DELETE detail statement, or a data statement causes an error condition. At least one blank space must precede and follow CHANGE. REPRO specifies that a member or a data set is copied in its entirety to a new master data set. At least one blank must precede and follow REPRO.

Members are logically deleted from a copy of a library by being omitted from a series of REPRO function statements within the same job step.

One sequential data set can be copied in a given job step. A sequential data set is logically deleted from a new volume by being omitted from a series of job steps which copy only the desired data sets to the new volume. If the NEW subparameter is coded in the EXEC statement, only the ADD function statement is permitted.

The format of the function statement is:

./[label]	{ADD|CHANGE|REPL|REPRO} [LIST=ALL] [,SEQFLD=(ddl|(ddl,ddl))] [,NEW=PO|PS] [,MEMBER=cccccccc] [,COLUMN=nn|1] [,UPDATE=INPLACE] [,INHDR=cccccccc] [,INTLR=cccccccc] [,OUTHDR=cccccccc] [,OUTTLR=cccccccc] [,TOTAL=(routinename,size)] [,NAME=cccccccc] [,LEVEL=hh] [,SOURCE=x] [,SSI=hhhhhhhh]

Function Restrictions

When UPDATE=INPLACE is specified:

- The SYSUT2 DD statement is not coded.
- The PARM parameter of the EXEC statement must imply or specify MOD.
- The NUMBER detail statement can be used to specify a renumbering operation.
- Data statements can be used to specify replacement information only.
- One CHANGE function statement and one UPDATE=INPLACE parameter are permitted per job step.
- No functions other than replacement, renumbering, and header label modification (via the LABEL statement) can be specified.
- Only replaced records are listed unless the entire data set is renumbered.
- System status information cannot be changed.

Within an existing logical record, the data in the field defined by the COLUMN parameter is replaced by data from a subsequent data statement, as follows:

1. IEBUPDTE matches a sequence number of a data statement with a sequence number of an existing logical record. In this manner, the COLUMN specification is applied to a specific logical record.

2. The information in the field within the data statement replaces the information in the field within the existing logical record. For example, COLUMN=40 indicates that columns 40 through 80 (assuming 80-byte logical records) of a subsequent data statement are to be used as replacement data for columns 40 through 80 of a logical record identified by a matching sequence number. (A sequence number in an existing logical record or data statement need not be within the defined field.)

The COLUMN specification applies to the entire function, with the exception of:

- Logical records deleted by a subsequent DELETE detail statement.
- Subsequent data statements not having a matching sequence number for an existing logical record.
- Data statements containing information to be inserted in the place of a deleted logical record or records.

Figure 100 shows the use of NEW, MEMBER, and NAME parameters for different input and output data set organizations.

Input Data Set Organization	Output Data Set Organization	Parameter Combinations
Partitioned	Partitioned	With an ADD function statement, use NAME to specify the name of the member to be placed in the partitioned data set defined by the SYSUT2 DD statement. If an additional name is required, an ALIAS statement can also be used. With a CHANGE, REPL, or REPRO function statement, use NAME to specify the name of the member within the partitioned data set defined by the SYSUT1 DD statement. If a different or additional name is desired for the member in the partitioned data set defined by the SYSUT2 DD statement, use an ALIAS statement also.
None	Partitioned (New)	With each ADD function statement, use NAME to assign a name for each member to be placed in the partitioned data set.

Figure 100 (Part 1 of 2). NEW, MEMBER, and NAME Parameter

Input Data Set Organization	Output Data Set Organization	Parameter Combinations
Partitioned	Sequential	With any function statement, use NAME to specify the name of the member in the partitioned data set defined by the SYSUT1 DD statement. Use NEW=PS to specify the change in organization from partitioned to sequential. (The name and file sequence number, if any, assigned to the output master data set are specified in the SYSUT2 DD statement.)
Sequential	Partitioned	With any function statement, use MEMBER to assign a name to the member to be placed in the partitioned data set defined by the SYSUT2 DD statement. Use NEW=PO to specify the change in organization from sequential to partitioned.

Figure 100 (Part 2 of 2). NEW, MEMBER, and NAME Parameter

Detail Statement

A detail statement is used with a function statement for certain applications, such as deleting or renumbering selected logical records. The NUMBER detail statement specifies, when coded with a CHANGE function statement, that the sequence number of one or more logical records is changed. It specifies, when coded with an ADD or REPL function statement, the sequence numbers to be assigned to the records within new or replacement members or data sets. When used with an ADD or REPL function statement, no more than one NUMBER detail statement is permitted for each ADD or REPL function statement. If NUMBER is coded, it must be preceded and followed by at least one blank.

The DELETE detail statement specifies, when coded with a CHANGE function statement, that one or more logical records are to be deleted from a member or data set. If DELETE is coded, it must be preceded and followed by at least one blank.

Logical records cannot be deleted in part; that is, a COLUMN parameter specification in a function statement is not applicable to records that are to be deleted. Each specific sequence number is handled only once in any single operation.

The format of a detail statement is:

./[label]	{NUMBER|DELETE}[SEQ1=cccccccc|ALL] [,SEQ2=cccccccc] [,NEW1=cccccccc] [,INCR=cccccccc] [,INSERT=YES]

Detail Restrictions

When INSERT=YES is coded:

- The SEQ1 parameter specifies the existing logical record after which the insertion is made. SEQ1=ALL cannot be coded.
- The SEQ2 parameter need not be coded.
- The NEW1 parameter assigns a sequence number to the first logical record to be inserted. If the parameter is alphameric, the SEQFLD=(ddl,ddl) parameter should be coded on the function statement.
- The INCR parameter is used to renumber as much as is necessary of the member or data set from the point of the first insertion; the member or data set is renumbered until an existing logical record is found whose sequence number is equal to or greater than the next sequence number to be assigned. If no such logical record is found, the entire member or data set is renumbered.
- Additional NUMBER detail statements, if any, must specify INSERT=YES. If a prior numbering operation renumbers the logical record specified in the SEQ1 parameter of a subsequent NUMBER detail statement, any NEW1 or INCR parameter specifications in the latter NUMBER detail statement are overridden. The prior increment value is used to assign the next successive sequence numbers. If a prior numbering operation does not renumber the logical record specified in the SEQ1 parameter of a subsequent NUMBER detail statement, the latter statement must contain NEW1 and INCR specifications.
- The block of data statements to be inserted must contain blank sequence numbers.
- The insert operation is terminated when a function statement, a detail statement, an end-of-file indication, or a data statement containing a sequence number is encountered.
- The SEQ1, SEQ2, and NEW1 parameters (with the exception of SEQ1=ALL) specify eight (maximum) alphameric characters. The INCR parameter specifies eight (maximum) numeric characters. Only the significant part of a numeric sequence number need be coded; for example, SEQ1=00000010 can be shortened to SEQ1=10. If, however, the numbers are alphameric, the alphabetic characters must be specified; for example, SEQ1=00ABC010 can be shortened to SEQ1=ABC010.

Data Statement

A data statement is used with a function statement, or with a function statement and a detail statement. It contains a logical record used as replacement data for an existing logical record, or new data to be incorporated in the output master data set.

Each data statement contains one logical record, which begins in the first column of the data statement. The length of the logical record is equal to the logical record length (LRECL) specified for the output master data set. Each logical record contains a sequence number to determine where the data is placed in the output master data set (except when INSERT=YES is specified).

When used with a CHANGE function statement, a data statement contains new or replacement data, as follows:

- If the sequence number in the data statement is identical to a sequence number in an existing logical record, the data statement replaces the existing logical record in the output master data set.
- If no corresponding sequence number is found within the existing records, the data statement is inserted in the proper collating sequence within the output master data set. (For proper execution of this function, all records in the old master data set must have a sequence number.)
- If a data statement with a sequence number is used and INSERT=YES was specified, the insert operation is terminated. IEBUPDTE will continue processing if this sequence number is at least equal to the next old master record (record following the referred to sequence record).

When used with an ADD or REPL function statement, a data statement contains new data to be placed in the output master data set.

Sequence numbers within the old master data set are assumed to be in ascending order. No validity checking of sequence numbers is performed for data statements or existing records.

Sequence numbers in data statements must be in the same relative position as sequence numbers in existing logical records. (Sequence numbers include leading zeros and are assumed to be in columns 73 through 80; if the numbers are in columns other than these, the length and relative position must be specified in a SEQFLD parameter within a preceding function statement.)

LABEL Statement

The LABEL statement indicates that the following data statements (called label data statements) are to be treated as user labels. These new user labels are placed on the output data set. The next function statement indicates to IEBUPDTE that the last label data statement of the group has been read.

The format of the LABEL statement is:

./[name]	LABEL

There can be no more than two LABEL statements per execution of IEBUPDTE. There can be no more than eight label data statements following any LABEL statement. The first 4 bytes of each 80-byte label data statement must contain "UHLn" or "UTLn," where n is 1 through 8, for input header or input trailer labels respectively, to conform to IBM standards for user labels. Otherwise, data management will overlay the data with the proper four characters.

When IEBUPDTE encounters a LABEL statement, it reads up to eight data statements and saves them for processing by user output label routines. If there are no such routines, the saved records are written by OPEN or CLOSE as user labels on the output data set. If there are user output label processing routines, IEBUPDTE passes a parameter list to the output label routines. (This parameter list is described fully in Appendix A, "Exit Routine Linkage" on page 438.) The label buffer contains a label data record which the user routine can process before the record is written as a label. If the user routine specifies (via return codes to IEBUPDTE) more entries than there are label data records, the label buffer will contain meaningless information for the remaining entries to the user routine.

The position of the LABEL statement in the SYSIN data set, relative to any function statements, indicates the type of user label that follows the LABEL statement:

- To create output header labels, place the LABEL statement and its associated label data statements before any function statements in the input stream. A function statement, other than LABEL, must follow the last label data statement of the group.
- To create output trailer labels, place the LABEL statement and its associated label data statements after any function statements in the input stream, but before the ENDUP statement. The ENDUP statement is not optional in this case. It must follow the last label data statement of the group if IEBUPDTE is to create output trailer labels.

When UPDATE=INPLACE is specified in a function statement, user input header labels can be updated by user routines, but input trailer and output labels cannot be updated by user routines. User labels cannot be added or deleted. User input header labels are made available to user routines by the label buffer address in the parameter list. (See Appendix C, "Processing User Labels" on page 446 for a complete discussion of the linkage between utility programs and user label processing routines.) The return codes when UPDATE=INPLACE is used differ slightly from the standard codes discussed in Appendix C as follows.

Codes	Meaning
00 (00 hex)	The system resumes normal processing; any additional user labels are ignored.
04 (04)	The system does not write the label. The next user label is read into the label buffer area and control is returned to the user's routine. If there are no more user labels, the system resumes normal processing.
08 (08)	The system writes the user labels from the label buffer area and resumes normal processing.
12 (0C)	The system writes the user label from the label buffer area, then reads the next input label into the label buffer area and returns control to the label processing routine. If there are no more user labels, the system resumes normal processing.

Figure 101. UPDATE=INPLACE Return Codes

If the user wants to examine the replaced labels from the old master data set, he must:

1. Specify an update of the old master by coding the UPDATE=INPLACE parameter in a function statement.
2. Include a LABEL statement in the input data set for either header or trailer labels.
3. Specify a corresponding user label routine.

If the above conditions are met, fourth and fifth parameter words will be added to the standard parameter list. The fourth parameter word is not now used; the fifth contains a pointer to the replaced label from the old master. In this case, the number of labels supplied in the SYSIN data set must not exceed the number of labels on the old master data set. If the user specifies, via return codes, more entries to the user's header label routine than there are labels in the input stream, the first parameter will point to the current header label on the old master data set for the remaining entries. In this case, the fifth parameter is meaningless.

ALIAS Statement

The ALIAS statement is used to create or retain an alias in an output (partitioned) directory. The ALIAS statement can be used with any of the function statements. Multiple aliases can be assigned to each member, up to a maximum of 16 aliases.

If an ALIAS statement specifies a name that already exists on the data set, the original TTR (track record) of that directory entry will be destroyed.

ALIAS must be preceded and followed by at least one blank. If ALIAS statements are used, they must follow the data statements, if any, in the input stream.

The format of the ALIAS statement is:

./[label]	ALIAS NAME=cccccccc

ENDUP Statement

An ENDUP statement is used to indicate the end of SYSIN input to this job step. It serves as an end-of-data indication if there is no other preceding delimiter statement. The ENDUP statement follows the last group of SYSIN control statements.

ENDUP must be preceded and followed by at least one blank. The ENDUP statement must follow the last label data statement if IEBUPDTE is used to create output trailer labels.

The format of the ENDUP statement is:

./[label]	ENDUP

Parameters	Applicable Control Statements	Description of Parameters
./	ADD REPL CHANGE REPRO NUMBER DELETE LABEL ALIAS ENDUP	**./** is required for each utility control statement and must appear in columns 1 and 2.
COLUMN	CHANGE	**COLUMN=**nn|**1** specifies, in decimal, the starting column of a data field within a logical record image. The field extends to the end of the image. Within an existing logical record, the data in the defined field is replaced by data from a subsequent data statement. See "Function Restrictions" on page 299 for restrictions on COLUMN.
INCR	NUMBER	**INCR=**cccccccc specifies an increment value used for assigning successive sequence numbers to new or replacement logical records, or specifies an increment value used for renumbering existing logical records.
INHDR	ADD REPL CHANGE REPRO	**INHDR=**cccccccc specifies the name of the user routine that handles any user input (SYSUT1) header labels. This parameter is valid only when a sequential data set is being processed.

Parameters	Applicable Control Statements	Description of Parameters
INSERT	CHANGE NUMBER	**INSERT=YES** specifies the insertion of a block of logical records. The records, which are data statements containing blank sequence numbers, are numbered and inserted in the output master data set. INSERT is valid only when coded with both a CHANGE function statement and a NUMBER detail statement. SEQ1, NEW1, and INCR are required on the first NUMBER detail statement. See "Detail Restrictions" on page 302 for more information on INSERT=YES.
INTLR	ADD REPL CHANGE REPRO	**INTLR=**cccccccc specifies the name of the user routine that handles any user input (SYSUT1) trailer labels. INTLR is valid only when a sequential data set is being processed, but not when UPDATE=INPLACE is coded.
label	ADD REPL CHANGE REPRO NUMBER DELETE LABEL ALIAS ENDUP	label specifies an optional label for the statement that begins in column 3 and extends no further than column 10.
LEVEL	ADD REPL CHANGE REPRO	**LEVEL=**hh specifies the change (update) level in hexadecimal (00-FF). The level number is recorded in the directory entry of the output member. This parameter is valid only when a member of a partitioned data set is being processed. LEVEL has no effect when SSI is specified.
LIST	ADD REPL CHANGE REPRO	**LIST=ALL** specifies that the SYSPRINT data set is to contain the entire updated member or data set and the control statements used in its creation. **Default:** For old data sets, if LIST is omitted, the SYSPRINT data set contains modifications and control statements only. If UPDATE was specified, the entire updated member is listed only when renumbering has been done. For new data sets, the entire member or data set and the control statements used in its creation are always written to the SYSPRINT data set.

Parameters	Applicable Control Statements	Description of Parameters
MEMBER	ADD REPL CHANGE REPRO	**MEMBER=**cccccccc specifies a name to be assigned to the member placed in the partitioned data set defined by the SYSUT2 DD statement. MEMBER is used only when SYSUT1 defines a sequential data set, SYSUT2 defines a partitioned data set, and NEW=PO is specified. Refer to Figure 100 on page 300 for the use of MEMBER with NEW.
NAME	ADD REPL CHANGE REPRO ALIAS	For the ALIAS statement: **NAME=**cccccccc specifies a 1- to 8-character alias name. For all other statements: **NAME=**cccccccc indicates the name of the member placed into the partitioned data set. The member name need not be specified in the DD statement itself. NAME must be provided to identify each input member. Refer to Figure 100 on page 300 for the use of NAME with NEW. This parameter is valid only when a member of a partitioned data set is being processed.
NEW	ADD REPL CHANGE REPRO	**NEW=PO|PS** specifies the organization of the old master data set and the organization of the updated output. NEW should not be specified unless the organization of the new master data set is different from the organization of the old master. Refer to Figure 100 on page 300 for the use of NEW with NAME and MEMBER. These values can be coded: **PO** specifies that the old master data set is a sequential data set, and that the updated output is to become a member of a partitioned data set. **PS** specifies that the old master data set is a partitioned data set, and that a member of that data set is to be converted into a sequential data set.
NEW1	NUMBER	**NEW1=**cccccccc specifies the first sequence number assigned to new or replacement data, or specifies the first sequence number assigned in a renumbering operation. A value specified in NEW1 must be greater than a value specified in SEQ1 (unless SEQ1=ALL is specified, in which case this rule does not apply).

Parameters	Applicable Control Statements	Description of Parameters
OUTHDR	ADD REPL CHANGE REPRO	**OUTHDR=**cccccccc specifies the name of the user routine that handles any user output (SYSUT2) header labels. OUTHDR is valid only when a sequential data set is being processed, but not when UPDATE=INPLACE is coded.
OUTTLR	ADD REPL CHANGE REPRO	**OUTTLR=**cccccccc specifies the name of the user routine that handles any user output (SYSUT2) trailer labels. OUTTLR is valid only when a sequential data set is being processed, but not when UPDATE=INPLACE is coded.
SEQ1	NUMBER DELETE	**SEQ1=**cccccccc|**ALL** specifies records to be renumbered, deleted, or assigned sequence numbers. These values can be coded: cccccccc specifies the sequence number of the first logical record to be renumbered or deleted. This value is not coded in a NUMBER detail statement that is used with an ADD or REPL function statement. When this value is used in an insert operation, it specifies the existing logical record after which an insert is to be made. It must not equal the number of a statement just replaced or added. Refer to the INSERT parameter for additional discussion. **ALL** specifies a renumbering operation for the entire member or data set. ALL is used only when a CHANGE function statement and a NUMBER detail statement are used. ALL must be coded if sequence numbers are to be assigned to existing logical records having blank sequence numbers. If ALL is not coded, all existing logical records having blank sequence numbers. copied directly to the output master data set. When ALL is coded: (1) SEQ2 need not be coded and (2) one NUMBER detail statement is permitted per function statement. Refer to the INSERT parameter for additional discussion.
SEQ2	NUMBER DELETE	**SEQ2=**cccccccc specifies the sequence number of the last logical record to be renumbered or deleted. SEQ2 is required on all DELETE detail statements. If only one record is to be deleted, the SEQ1 and SEQ2 specifications must be identical. SEQ2 is not coded in a NUMBER detail statement that is used with an ADD or REPL function statement.

Parameters	Applicable Control Statements	Description of Parameters
SEQFLD	ADD REPL CHANGE REPRO	**SEQFLD=**ddl|(ddl,ddl) ddl specifies, in decimal, the starting column (up to column 80) and length (8 or less) of sequence numbers within existing logical records and subsequent data statements. Note that the starting column specification (dd) plus the length (l) cannot exceed the logical record length (LRECL) plus 1. Sequence numbers on incoming data statements and existing logical records must be padded to the left with enough zeros to fill the length of the sequence field. (ddl,ddl) may be used when an alphameric sequence number generation is required. The first ddl specifies the sequence number columns as above. The second ddl specifies, in decimal, the starting column (up to column 80) and length (8 or less) of the numeric portion of the sequence numbers in subsequent NUMBER statements. This information is used to determine which portion of the sequence number specified by the NEW1 parameter may be increased and which portion(s) should be copied to generate a new sequence number for inserted or renumbered records. The numeric columns must fall within the sequence number columns specified (or defaulted) by the first ddl. **Default:** 738 is assumed, that is, an 8-byte sequence number beginning in column 73. Therefore, if existing logical records and subsequent data statements have sequence numbers in columns 73 through 80, this keyword need not be coded.
SOURCE	ADD REPL CHANGE REPRO	**SOURCE=**x specifies user modifications when the x value is 0, or IBM modifications when the x value is 1. The source is recorded in the directory entry of the output member. This parameter is valid only when a member of a partitioned data set is being processed. SOURCE has no effect when SSI is specified.

Parameters	Applicable Control Statements	Description of Parameters
SSI	ADD REPL CHANGE REPRO	**SSI=**hhhhhhhh specifies eight hexadecimal characters of system status information (SSI) to be placed in the directory of the new master data set as four packed decimal bytes of user data. This parameter is valid only when a member of a partitioned data set is being processed. SSI overrides any LEVEL or SOURCE parameter given on the same function statement.
TOTAL	ADD REPL CHANGE REPRO	**TOTAL=(**routinename,size**)** specifies that exits to a user's routine are to be provided prior to writing each record. This parameter is valid only when a sequential data set is being processed. These values are coded: routinename specifies the name of the user's totaling routine. size specifies the number of bytes required for the user's data. The size should not exceed 32K, nor be less than 2 bytes. In addition, the keyword OPTCD=T must be specified for the SYSUT2 (output) DD statement. Refer to Appendix A, "Exit Routine Linkage" on page 438 for a discussion of linkage conventions for user routines.
UPDATE	CHANGE	**UPDATE=INPLACE** specifies that the old master data set is to be updated within the space it actually occupies. The old master data set must reside on a direct access device. UPDATE=INPLACE is valid only when coded with CHANGE. No other function statements (ADD, REPL, REPRO) may be in the same job step. See "Function Restrictions" on page 299 for restrictions on using UPDATE=INPLACE. See "LABEL Statement" on page 303 for information on updating user input header labels.

IEBUPDTE EXAMPLES

The following examples illustrate some of the uses of IEBUPDTE. Figure 102 can be used as a quick-reference guide to IEBUPDTE examples. The numbers in the "Example" column point to examples that follow.

Operation	Data Set Organization	Device	Comments	Example
ADD and REPL	Partitioned	Disk	SYSUT1 and SYSUT2 DD statements define the same data set. A JCL procedure residing in the control data set is stored as a new member of a procedure library (PROCLIB). Another JCL procedure, also in the IEBUPDTE control data set, is to replace an existing member in PROCLIB.	1
CREATE a partitioned library	Partitioned	Disk	Input data is in the control data set. Output partitioned data set is to contain three members.	2
CREATE a partitioned data set	Partitioned	Disk	Input from control data set and from existing partitioned data set. Output partitioned data set is to contain three members.	3
UPDATE INPLACE and renumber	Partitioned	Disk	Input data set is considered to be the output data set as well; therefore, no SYSUT2 DD statement is required.	4
CREATE and DELETE	Partitioned, Sequential	Disk and Tape	Sequential master is created from partitioned disk input. Selected records are to be deleted. Blocked output.	5
CREATE, DELETE, and UPDATE	Sequential, Partitioned	Tape and Disk	Partitioned data set is created from sequential input. Records are to be deleted and updated. Sequence numbers in columns other than 73 through 80. One member is placed in the output data set.	6
INSERT	Partitioned	Disk	Block of logical records is inserted into an existing member. SYSUT1 and SYSUT2 DD statements define the same data set.	7
INSERT	Partitioned	Disk	Two blocks of logical records are to be inserted into an existing member. SYSUT1 and SYSUT2 DD statements define the same data set. Sequence numbers are alphameric.	8
CREATE	Sequential	Card Reader and Disk	Sequential data set with user labels is to be created from card input.	9

Figure 102 (Part 1 of 2). IEBUPDTE Example Directory

Operation	Data Set Organization	Device	Comments	Example
COPY	Sequential	Disk	Sequential data set is copied from one direct access volume to another; user labels can be processed by exit routines.	10
CREATE	Partitioned	Disk	Create a new generation.	11

Figure 102 (Part 2 of 2). IEBUPDTE Example Directory

Examples that use **disk** or **tape** in place of actual device numbers must be changed before use. See "DASD and Tape Device Support" on page 3 for valid device number notation.

IEBUPDTE EXAMPLE 1

In this example, two procedures are to be placed in the cataloged procedure library, SYS1.PROCLIB. The example assumes that the two procedures can be accommodated within the space originally allocated to the procedure library.

```
//UPDATE   JOB  09#660,SMITH
//         EXEC PGM=IEBUPDTE,PARM=MOD
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=SYS1.PROCLIB,DISP=OLD
//SYSUT2   DD   DSNAME=SYS1.PROCLIB,DISP=OLD
//SYSIN    DD   DATA
./         ADD  LIST=ALL,NAME=ERASE,LEVEL=01,SOURCE=0
./     NUMBER   NEW1=10,INCR=10
//ERASE    EXEC PGM=IEBUPDTE
//DD1      DD   UNIT=disk,DISP=(OLD,KEEP),VOLUME=SER=111111
//SYSPRINT DD   SYSOUT=A
./        REPL  LIST=ALL,NAME=LISTPROC
./     NUMBER   NEW1=10,INCR=10
//LIST     EXEC PGM=IEBGENER
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DISP=SHR,
//              DSN=SYS1.PROCLIB(&MEMBER)
//SYSUT2   DD   SYSOUT=A,
//              DCB=(RECFM=F,BLKSIZE=80)
//SYSIN    DD   DATA
(Data statements)
./       ENDUP
/*
```

The control statements are discussed below:

- SYSUT1 and SYSUT2 DD define the SYS1.PROCLIB data set, which is assumed to be cataloged.
- SYSIN DD defines the control data set, which follows in the input stream. The data set contains the utility control statements and the data to be placed in the procedure library.
- The ADD function statement indicates that records (data statements) in the control data set are to be placed in the output. The newly created procedure, ERASE, is listed in the message data set.

The ADD function will not take place if a member named ERASE already exists in the new master data set referenced by SYSUT2.

- The first NUMBER detail statement indicates that the new and replacement procedures are to be assigned sequence numbers. The first record of each procedure is assigned sequence number 10; the next record is assigned sequence number 20, and so on.
- The REPL function statement indicates that records (data statements) in the control data set are to replace an already existing member. The member is stored in the new master data set referenced by SYSUT2. The REPL function will only take place if a member named LISTPROC already exists in the old master data set referenced by SYSUT1.
- The ERASE EXEC statement marks the beginning of the first new procedure.
- The REPL function statement indicates that records (data statements) in the control data set are to replace an already existing member. The member is stored in the new master data set referenced by SYSUT2. The REPL function will only take place if a member named LISTPROC already exists in the old master data set referenced by SYSUT1.
- The second NUMBER detail statement is a duplicate of the first.
- The LIST EXEC statement marks the beginning of the second new procedure.
- The ENDUP statement marks the end of the SYSIN DD input data.

IEBUPDTE EXAMPLE 2

In this example, a three-member partitioned library is created. The input data is contained solely in the control data set.

```
//UPDATE   JOB  09#770,SMITH
//         EXEC PGM=IEBUPDTE,PARM=NEW
//SYSPRINT DD   SYSOUT=A
//SYSUT2   DD   DSNAME=OUTLIB,UNIT=disk,DISP=(NEW,KEEP),
//              VOLUME=SER=111112,SPACE=(TRK,(50,,10)),
//              DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//SYSIN    DD   DATA
./        ADD   NAME=MEMB1,LEVEL=00,SOURCE=0,LIST=ALL
(Data statements, sequence numbers in columns 73 through 80)
./        ADD   NAME=MEMB2,LEVEL=00,SOURCE=0,LIST=ALL
(Data statements, sequence numbers in columns 73 through 80)
./        ADD   NAME=MEMB3,LEVEL=00,SOURCE=0,LIST=ALL
(Data statements, sequence numbers in columns 73 through 80)
./      ENDUP
/*
```

The control statements are discussed below:

- SYSUT2 DD defines the new partitioned master, OUTLIB. Enough space is allocated to allow for subsequent modifications without creating a new master data set.

- SYSIN DD defines the control data set, which follows in the input stream. The data set contains the utility control statements and the data to be placed as three members in the output partitioned data set.
- The ADD function statements indicate that subsequent data statements are to be placed as members in the output partitioned data set. Each ADD function statement specifies a member name for subsequent data and indicates that the member and control statement is listed in the message data set.
- The data statements contain the data to be placed in each output partitioned data set.
- ENDUP signals the end of control data set input.

Because sequence numbers (other than blank numbers) are included within the data statements, no NUMBER detail statements are included in the example.

IEBUPDTE EXAMPLE 3

In this example, a three-member, partitioned data set (NEWMCLIB) is created. The data set will contain:

- Two members (ATTACH and DETACH) copied from an existing partitioned data set (SYS1.MACLIB).
- A new member (EXIT), which is contained in the control data set.

```
//UPDATE   JOB  09#770,SMITH
//        EXEC PGM=IEBUPDTE,PARM=MOD
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=SYS1.MACLIB,DISP=SHR,UNIT=disk
//SYSUT2   DD   DSNAME=NEWMCLIB,VOLUME=SER=111112,UNIT=disk,
//              DISP=(NEW,KEEP),SPACE=(TRK,(100,,10)),
//              DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//SYSIN    DD   DATA
./      REPRO   NAME=ATTACH,LEVEL=00,SOURCE=1,LIST=ALL
./      REPRO   NAME=DETACH,LEVEL=00,SOURCE=1,LIST=ALL
./        ADD   NAME=EXIT,LEVEL=00,SOURCE=1,LIST=ALL
./     NUMBER   NEW1=10,INCR=100

(Data records for EXIT member)

./      ENDUP
/*
```

The control statements are discussed below:

- SYSUT1 DD defines the input partitioned data set SYS1.MACLIB, which is assumed to be cataloged.
- SYSUT2 DD defines the output partitioned data set NEWMCLIB. Enough space is allocated to allow for subsequent modifications without creating a new master data set.
- SYSIN DD defines the control data set, which follows in the input stream.
- The REPRO function statements identify the existing input members (ATTACH and DETACH) to be copied onto the output data set. These members are also listed in the message data set (since LIST=ALL is specified).

- The ADD function statement indicates that records (subsequent data statements) are to be placed as a member in the output partitioned data set, called EXIT. The data statements are to be listed in the message data set.
- The NUMBER detail statement assigns sequence numbers to the data statements. (The data statements contain blank sequence numbers in columns 73 through 80.) The first record of the output member is assigned sequence number 10; subsequent records are incremented by 100.
- ENDUP signals the end of SYSIN data.

Note that the three named input members (ATTACH, DETACH, and EXIT) do not have to be specified in the order of their collating sequence in the old master.

IEBUPDTE EXAMPLE 4

In this example, a member (MODMEMB) is updated within the space it actually occupies. Two existing logical records are replaced, and the entire member is renumbered.

```
//UPDATE   JOB  09#770,SMITH
//         EXEC PGM=IEBUPDTE,PARM=MOD
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=PDS,UNIT=disk,DISP=(OLD,KEEP),
//              VOLUME=SER=111112
//SYSIN    DD   *
./     CHANGE   NAME=MODMEMB,LIST=ALL,UPDATE=INPLACE
./     NUMBER   SEQ1=ALL,NEW1=10,INCR=5

(Data statement 1, sequence number 00000020)
(Data statement 2, sequence number 00000035)

/*
```

The control statements are discussed below:

- SYSUT1 DD defines the partitioned data set that is updated in place. (Note that the member name need not be specified in the DD statement.)
- SYSIN DD defines the control data set, which follows in the input stream.
- The CHANGE function statement indicates the name of the member to be updated (MODMEMB) and specifies the UPDATE=INPLACE operation. The entire member is listed in the message data set. Note that, as renumbering is being done, and since UPDATE=INPLACE was specified, the listing would have been provided even if the LIST=ALL parameter had not been specified. See the LIST parameter for more information.
- The NUMBER detail statement indicates that the entire member is to be renumbered, and specifies the first sequence number to be assigned and the increment value (5) for successive sequence numbers.
- The data statements replace existing logical records having sequence numbers of 20 and 35.

IEBUPDTE EXAMPLE 5

In this example, a new master sequential data set is created from partitioned input and selected logical records are deleted.

```
//UPDATE   JOB  09#770,SMITH
//         EXEC PGM=IEBUPDTE,PARM=MOD
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=PARTDS,UNIT=disk,DISP=(OLD,KEEP),
//              VOLUME=SER=111112
//SYSUT2   DD   DSNAME=SEQDS,UNIT=tape,LABEL=(2,SL),
//              DISP=(,KEEP),VOLUME=SER=001234,
//              DCB=(RECFM=FB,LRECL=80,BLKSIZE=2000)
//SYSIN    DD   *
./     CHANGE   NEW=PS,NAME=OLDMEMB1

(Data statement 1, sequence number 00000123)

./     DELETE   SEQ1=223,SEQ2=246

(Data statement 2, sequence number 00000224)

/*
```

The control statements are discussed below:

- SYSUT1 DD defines the input partitioned data set PARTDS, which resides on a disk volume.
- SYSUT2 DD defines the output sequential data set, SEQDS. The data set is written as the second data set on a tape volume.
- SYSIN DD defines the control data set, which follows in the input stream.
- CHANGE identifies the input member (OLDMEMB1) and indicates that the output is a sequential data set (NEW=PS).
- The first data statement replaces the logical record whose sequence number is identical to the sequence number in the data statement (00000123). If no such logical record exists, the data statement is incorporated in the proper sequence within the output data set.
- The DELETE detail statement deletes logical records having sequence numbers from 223 through 246, inclusive.
- The second data statement is inserted in the proper sequence in the output data set, since no logical record with the sequence number 224 exists (it was deleted in the previous statement).

Note that only one member can be used as input when converting to sequential organization.

IEBUPDTE EXAMPLE 6

In this example, a member of a partitioned data set is created from sequential input and existing logical records are updated.

```
                                                                      72
//UPDATE   JOB  09#770,SMITH
//         EXEC PGM=IEBUPDTE,PARM=MOD
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=OLDSEQDS,UNIT=tape,
//              DISP=(OLD,KEEP),VOLUME=SER=001234
//SYSUT2   DD   DSNAME=NEWPART,UNIT=disk,DISP=(,KEEP),
//              VOLUME=SER=111112,SPACE=(TRK,(10,5,5)),
//              DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//SYSIN    DD   *
./     CHANGE   NEW=PO,MEMBER=PARMEM1,LEVEL=01,                      C
./              SEQFLD=605,COLUMN=40,SOURCE=0

(Data statement 1, sequence number 00020)

./     DELETE   SEQ1=220,SEQ2=250

(Data statement 2, sequence number 00230)
(Data statement 3, sequence number 00260)

./      ALIAS   NAME=MEMB1
/*
```

The control statements are discussed below:

- SYSUT1 DD defines the input sequential data set (OLDSEQDS). The data set resides on a tape volume.
- SYSUT2 DD defines the output partitioned data set (NEWPART). Enough space is allocated to provide for members that might be added in the future.
- SYSIN DD defines the control data set, which follows in the input stream.
- The CHANGE function statement identifies the output member (PARMEM1) and indicates that a conversion from sequential input to partitioned output is made. The SEQFLD parameter indicates that a 5-byte sequence number is located in columns 60 through 64 of each data statement. The COLUMN=40 parameter specifies the starting column of a field (within subsequent data statements) from which replacement information is obtained. SOURCE=0 indicates that the replacement information is provided by the user.
- The first data statement is used as replacement data. Columns 40 through 80 of the statement replace columns 40 through 80 of the corresponding logical record. If no such logical record exists, the entire card image is inserted in the output data set member.
- The DELETE detail statement deletes all of the logical records having sequence numbers from 220 through 250.
- The second data statement, whose sequence number falls within the range specified in the DELETE detail statement above, is incorporated in its entirety in the output data set member.
- The third data statement, which is beyond the range of the DELETE detail statement, is treated in the same manner as the first data statement.
- ALIAS assigns the alias name MEMB1 to the output data set member PARMEM1.

IEBUPDTE EXAMPLE 7

In this example, a block of three logical records is inserted into an existing member, and the updated member is placed in the existing partitioned data set.

```
//UPDATE   JOB  09#770,SMITH
//        EXEC PGM=IEBUPDTE,PARM=MOD
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=PDS,UNIT=disk,DISP=(OLD,KEEP),
//              VOLUME=SER=111112
//SYSUT2   DD   DSNAME=PDS,UNIT=disk,DISP=(OLD,KEEP),
//              VOLUME=SER=111112
//SYSIN    DD   *
./     CHANGE   NAME=RENUM,LIST=ALL,LEVEL=01,SOURCE=0
./     NUMBER   SEQ1=15,NEW1=20,INCR=5,INSERT=YES

(Data statement 1)
(Data statement 2)
(Data statement 3)

/*
```

The control statements are discussed below:

- SYSUT1 and SYSUT2 DD define the partitioned data set (PDS).
- SYSIN DD defines the control data set, which follows in the input stream.
- The CHANGE function statement identifies the input member RENUM. The entire member is listed in the message data set.
- The NUMBER detail statement specifies the insert operation and controls the renumbering operation as described below.
- The data statements are the logical records to be inserted. (Sequence numbers are assigned when the data statements are inserted.)

In this example, the existing logical records have sequence numbers 10, 15, 20, 25, and 30. Sequence numbers are assigned by the NUMBER detail statement, as follows:

1. Data statement 1 is assigned sequence number 20 (NEW1=20) and inserted after existing logical record 15 (SEQ1=15).
2. Data statements 2 and 3 are assigned sequence numbers 25 and 30 (INCR=5) and are inserted after data statement 1.
3. Existing logical records 20, 25, and 30 are assigned sequence numbers 35, 40, and 45, respectively.

Figure 103 on page 320 shows existing sequence numbers, data statements inserted, and the resultant new sequence numbers.

Sequence Numbers and Data Statements Inserted	New Sequence Numbers
10	10
15	15
Data statement 1	20
Data statement 2	25
Data statement 3	30
20	35
25	40
30	45

Figure 103. Example of Reordered Sequence Numbers

IEBUPDTE EXAMPLE 8

In this example, two blocks (three logical records per block) are inserted into an existing member, and the member is placed in the existing partitioned data set. A portion of the output member is also renumbered.

```
//UPDATE   JOB  09#770,SMITH
//         EXEC PGM=IEBUPDTE,PARM=MOD                              72
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=PDS,UNIT=disk,DISP=(OLD,KEEP),
//              VOLUME=SER=111112
//SYSUT2   DD   DSNAME=PDS,UNIT=disk,DISP=(OLD,KEEP),
//              VOLUME=SER=111112
//SYSIN    DD   *
./     CHANGE   NAME=RENUM,LIST=ALL,LEVEL=01,SOURCE=0,            C
./              SEQFLD=(765,783)
./     NUMBER   SEQ1=AA015,NEW1=AA020,INCR=5,INSERT=YES

(Data statement 1)
(Data statement 2)
(Data statement 3)

./     NUMBER   SEQ1=AA030,INSERT=YES

(Data statement 4)
(Data statement 5)
(Data statement 6)
(Data statement 7, sequence number AA035)

/*
```

The control statements are discussed below:

- SYSUT1 and SYSUT2 DD define the partitioned data set PDS.
- SYSIN DD defines the control data set, which follows in the input stream.
- The CHANGE function statement identifies the input member RENUM. The entire member is listed in the message data set.
- The NUMBER detail statements specify the insert operations (INSERT=YES) and control the renumbering operation as described below.

- Data statements 1, 2, 3, and 4, 5, 6 are the blocks of logical records to be inserted. Because they contain blank sequence numbers, sequence numbers are assigned when the data statements are inserted.

- Data statement 7, since it contains a sequence number, terminates the insert operation. The sequence number is identical to the number on the next record in the old master data set; consequently, data statement 7 will replace the equally numbered old master record in the output data set.

The existing logical records in this example have sequence numbers AA010, AA015, AA020, AA025, AA030, AA035, AA040, AA045, AA050, BB010, and BB015. The insert and renumbering operations are performed as follows:

1. Data statement 1 is assigned sequence number AA020 (NEW1=AA020) and inserted after existing logical record AA015 (SEQ1=AA015).

2. Data statements 2 and 3 are assigned sequence numbers AA025 and AA030 (INCR=5) and are inserted after data statement 1.

3. Existing logical records AA020, AA025, and AA030 are assigned sequence numbers AA035, AA040, and AA045, respectively.

4. Data statement 4 is assigned sequence number AA050 and inserted. (The SEQ1=AA030 specification in the second NUMBER statement places this data statement after existing logical record AA030, which has become logical record AA045.)

5. Data statements 5 and 6 are assigned sequence numbers AA055 and AA060 and are inserted after data statement 4.

6. Existing logical record AA035 is replaced by data statement 7, which is assigned sequence number AA065.

7. The remaining logical records in the member are renumbered until logical record BB010 is encountered. Because this record has a sequence number higher than the next number to be assigned, the renumbering operation is terminated.

Figure 104 on page 322 shows existing sequence numbers, data statements inserted, and the new sequence numbers. Note that the sequence numbers are alphameric.

Sequence Numbers and Data Statements Inserted	New Sequence Numbers
AA010	AA010
AA015	AA015
Data statement 1	AA020
Data statement 2	AA025
Data statement 3	AA030
AA020	AA035
AA025	AA040
AA030	AA045
Data statement 4	AA050
Data statement 5	AA055
Data statement 6	AA060
Data statement 7	AA065
AA035	AA065
AA040	AA070
AA045	AA070
AA050	AA075
BB010	BB010
BB015	BB015

Figure 104. Reordered Sequence Numbers.

IEBUPDTE EXAMPLE 9

In this example, IEBUPDTE is used to create a sequential data set from card input. User header and trailer labels, also from the input stream, are placed on this sequential data set.

```
//LABEL    JOB  ,MSGLEVEL=1
//CREATION EXEC PGM=IEBUPDTE,PARM=NEW
//SYSPRINT DD   SYSOUT=A
//SYSUT2   DD   DSNAME=LABEL,VOLUME=SER=123456,UNIT=disk,
//              DISP=(NEW,KEEP),LABEL=(,SUL),
//              SPACE=(TRK,(15,3))
//SYSIN    DD   *
./      LABEL

(First header label)
.
.
.
(Last header label)

./        ADD   LIST=ALL,OUTHDR=ROUTINE1,OUTTLR=ROUTINE2

(First input data record)
.
.
.
(Last input data record)

./      LABEL

(First trailer label)
.
.
.
(Last trailer label)

./      ENDUP
/*
```

The control statements are discussed below:

- SYSUT2 DD defines and allocates space for the output sequential data set, called LABEL, which resides on a disk volume.
- SYSIN DD defines the control data set, which follows in the input stream. (This control data set includes the sequential input data set and the user labels, which are on cards.)
- The first LABEL statement identifies the 80-byte card images in the input stream which will become user header labels. (They can be modified by the user's header-label processing routine specified on the ADD function statement.)
- The ADD function statement indicates that the data statements that follow are placed in the output data set. The newly created data set is listed in the message data set. User output header and output trailer routines are to be given control prior to the writing of header and trailer labels.
- The second LABEL statement identifies the 80-byte card images in the input stream which will become user trailer labels. (They can be modified by the user's trailer-label processing routine specified on the ADD function statement.)
- ENDUP signals the end of the control data set.

IEBUPDTE EXAMPLE 10

In this example, IEBUPDTE is used to copy a sequential data set from one DASD volume to another. User labels are processed by user exit routines.

```
//LABELS   JOB  ,MSGLEVEL=1
//         EXEC PGM=IEBUPDTE,PARM=(MOD,,MMMMMM)
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=OLDMAST,DISP=OLD,LABEL=(,SUL),
//              VOLUME=SER=111111,UNIT=disk
//SYSUT2   DD   DSNAME=NEWMAST,DISP=(NEW,KEEP),LABEL=(,SUL),
//              UNIT=disk,VOLUME=SER=XB182,
//              SPACE=(TRK,(5,10))
//SYSIN    DD   DSNAME=INPUT,DISP=OLD,LABEL=(,SUL),
//              VOLUME=SER=222222,UNIT=disk
/*
```

The control statements are discussed below:

- SYSUT1 DD defines the input sequential data set, called OLDMAST, which resides on a disk volume.
- SYSUT2 DD defines the output sequential data set, called NEWMAST, which will reside on a disk volume.
- SYSIN DD defines the control data set. The contents of this disk-resident data set in this example are:

```
./      REPRO   LIST=ALL,INHDR=SSSSSS,INTLR=TTTTTT,          C
./              OUTHDR=XXXXXX,OUTTLR=YYYYYY
./      ENDUP
```

- The REPRO function statement indicates that the existing input sequential data set is copied to the output data set. This output data set is listed on the message data set. The user's label processing routines are to be given control when header or trailer labels are encountered on either the input or the output data set.
- ENDUP indicates the end of the control data set.

IEBUPDTE EXAMPLE 11

In this example, a partitioned generation data set consisting of three members is used as source data in the creation of a new generation data set. IEBUPDTE is also used to add a fourth member to the three source members and to number the new member. The resultant data set is cataloged as a new generation data set.

```
//          JOB
//          EXEC PGM=IEBUPDTE,PARM=MOD
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=A.B.C(0),DISP=OLD
//SYSUT2   DD   DSNAME=A.B.C(+1),DISP=(,CATLG),UNIT=disk,
//              VOLUME=SER=111111,SPACE=(TRK,(100,10,10)),
//              DCB=(RECFM=FB,LRECL=80,BLKSIZE=800)
//SYSIN    DD   DATA
./  REPRO    NAME=MEM1,LEVEL=00,SOURCE=0,LIST=ALL
./  REPRO    NAME=MEM2,LEVEL=00,SOURCE=0,LIST=ALL
./  REPRO    NAME=MEM3,LEVEL=00,SOURCE=0,LIST=ALL
./  ADD      NAME=MEM4,LEVEL=00,SOURCE=0,LIST=ALL
./  NUMBER   NEW1=10,INCR=5

(data cards comprising MEM4)

./  ENDUP
/*
```

The control statements are discussed below:

- SYSUT1 DD defines the latest generation data set, which is used as source data.
- SYSUT2 DD defines the new generation data set, which is created from the source generation data set and from an additional member included as input and data.
- SYSIN DD defines the control data set, which follows in the input stream.
- The REPRO function statements reproduce the named source members in the output generation data set.
- The ADD function statement specifies that the data cards following the input stream be included as MEM4.
- The NUMBER detail statement indicates that the new member is to have sequence numbers assigned in columns 73 through 80. The first record is assigned sequence number 10. The sequence number of each successive record is increased by 5.
- ENDUP signals the end of input card data.

This example assumes that a model data set control block (DSCB) exists on the catalog volume on which the generation data group index was built.

IEHATLAS PROGRAM

IEHATLAS is a system utility used with direct access devices when a defective track is indicated by a data check or missing address marker condition.

IEHATLAS can be used to locate and assign an alternate track to replace the defective track. Usable data records on the defective track are retrieved and transferred to the alternate track. A replacement for the bad record is created from data supplied by the user and placed on the alternate track.

In a simple application, IEHATLAS is used as a separate job after an abnormal termination of a problem program. Input data necessary for execution of IEHATLAS—the address of the defective track and replacement records—may be obtained from the dump and from backup data.

A more complex use of IEHATLAS may involve the preparation of a user's SYNAD routine, which reconstructs the necessary input data and invokes IEHATLAS dynamically.

When IEHATLAS is invoked, it attempts to write on the defective track. If the subsequent read-back check indicates that the attempt was successful, a message is issued on the SYSOUT device. If not, a supervisor call routine (SVC 86) is entered automatically.

The SVC routine locates and assigns an alternate track. (If a defective track already has an alternate and an error occurs on that alternate, the SVC routine assigns the next available alternate.) All of the valid data records on the defective track are retrieved and transferred to the alternate track. The input record is written on the alternate track in the correct position to recover from the previous error.

When a READ error occurs and a complete recovery is desired, see Data Facility Data Set Services: User's Guide and Reference for information on how to produce a listing of error data on a track. Using this data, the input data record for IEHATLAS can be created. The replace function can then be performed by executing IEHATLAS.

IEHATLAS supports all current DASD, as listed under "DASD and Tape Device Support" on page 3 except the MSS staging packs and virtual volumes.

INPUT AND OUTPUT

IEHATLAS uses the following input:

- A description of the count field of the invalid record on a defective track, specifying the cylinder, track, record, key, and data length (in hexadecimal notation).
- An indication if WRITE special is needed.
- A valid copy (in hexadecimal notation) of the bad record.

IEHATLAS produces as output:

- A message, issued on the SYSOUT device, containing the user's control information, the input record, and diagnostics.
- The input record, written on either the original (defective) track or on an alternate track containing the usable data taken from the defective track.

- The return parameter list (specifying a maximum of three error record numbers in hexadecimal when an unrecoverable error occurs).

RETURN CODES

IEHATLAS returns one of the following codes in register 15 when processing stops.

Codes	Meaning
00 (00 hex)	Successful completion; IEHATLAS has assigned the data to an alternate track.
04 (04)	The device does not have software-assignable alternate tracks.
08 (08)	All the alternate tracks for the device have been assigned.
12 (0C)	The requested main storage space is not available.
16 (10)	There was an I/O error in the alternate track assignment after N attempts at assignment (where N=10% of the assignable alternate tracks for this device).
20 (14)	The error is a condition other than a data check or missing address marker.
24 (18)	There is an error in the Format 4 DSCB that prevents IEHATLAS from reading it.
28 (1C)	The user-specified error record is the Format 4 DSCB, which IEHATLAS cannot handle because the alternate track information is unreliable.
32 (20)	IEHATLAS cannot handle the error found in the count field of the last record on the track.
36 (24)	There are errors in the home address or in record zero.
40 (28)	IEHATLAS found one or more errors in record(s) and assigned an alternate track: 1. There was an error on an end-of-file record, 2. IEHATLAS encountered an error in the count field, 3. There were errors in more than three count fields.
48 (30)	IEHATLAS found no errors on the track specified and so assigned no alternate track.
52 (34)	Because of an I/O error, IEHATLAS cannot reexecute the user's channel program successfully.
56 (38)	The system does not support track overflow.
60 (3C)	The track address provided does not belong to the indicated data set.

Figure 105. IEHATLAS Return Codes

CONTROL

IEHATLAS is controlled by job control statements and utility control statements. The job control statements are used to execute or invoke IEHATLAS and to define the data sets used and produced by IEHATLAS.

A utility control statement is used to specify whether the bad record is part of the volume table of contents. It is also used to indicate whether or not the WRITE special CCW command is to be used for track overflow records.

JOB CONTROL STATEMENTS

Figure 106 shows the job control statements for IEHATLAS.

Statement	Use
JOB	Initiates the job.
EXEC	Specifies the program name (PGM=IEHATLAS) or, if the job control statements reside in a procedure library, the procedure name.
SYSPRINT DD	Defines a sequential data set that contains the output messages issued by IEHATLAS.
SYSUT1 DD	Defines the data set that contains the bad record.
SYSIN DD	Defines the control data set, which contains the utility control statement and a copy of the corrected version of the bad record.

Figure 106. Job Control Statements for IEHATLAS

The block size for the SYSPRINT data set must be a multiple of 121. Any blocking factor can be specified.

The block size for the SYSIN data set must be a multiple of 80. Any blocking factor can be specified.

DISP=SHR must not be coded on the SYSUT1 DD statement.

UTILITY CONTROL STATEMENTS

Figure 107 shows the utility control statements for IEHATLAS.

Statement	Use
TRACK	Specifies that an alternate track is to be assigned for a track that does not contain VTOC records.
VTOC	Specifies that an alternate track is to be assigned for a track that contains VTOC records.

Figure 107. Utility Control Statements for IEHATLAS

Input data (consisting of the hexadecimal replacement record) begins in column 1 immediately following the utility control data. Input data may continue through column 80. As many cards as necessary may be used to contain the replacement record. All columns (1 through 80) are used on the additional cards.

IEHATLAS is designed to replace an error record with a copy of that record. It cannot be used to replace a record with another of a different key and/or data length.

An end-of-file record cannot be changed; therefore, input for key and/or data fields are ignored.

Continuation requirements for the utility control statements are described in "Continuing Utility Control Statements" on page 5.

TRACK Statement

The TRACK statement is used to identify a defective track which does **not** contain VTOC records (that is, the defective record is not included in the volume table of contents).

The TRACK statement must not begin in column 1.

The format of the TRACK statement is:

VTOC Statement

The VTOC statement is used to identify a defective track which contains VTOC records (that is, the defective record is included in the volume table of contents).

The VTOC statement must not begin in column 1.

The format of the VTOC statement is:

Parameters	Applicable Control Statements	Description of Parameters
bbbb	TRACK VTOC	bbbb This number must be all zeros.
cccc	TRACK VTOC	cccc is the hexadecimal number of the cylinder in which the defective track was found.
dddd	TRACK VTOC	dddd is the hexadecimal data length of the bad record. (When a WRITE special command is used, dddd is the length of the record segment.) dddd must not exceed the data length specified in the count field of the defective record.
hhhh	TRACK VTOC	hhhh is the defective track number, in hexadecimal.
rrkk	TRACK VTOC	rrkk is the record number and key length for the bad record, in hexadecimal. kk must not exceed the key length specified in the count field of the defective record.
S	TRACK	S is an optional byte of EBCDIC information that specifies that the WRITE special command is to be used (when the last record on the track overflows and must be completed elsewhere).

IEHATLAS EXAMPLES

The following examples illustrate some of the uses of IEHATLAS. Figure 108 can be used as a quick-reference guide to IEHATLAS examples. The numbers in the "Example" column point to examples that follow.

Operation	Comments	Example
Get Alternate Track	Write special is included because of a track overflow condition.	1
Get Alternate Track	Alternate track assigned for a bad end-of-file record.	2
Get Alternate Track	Alternate track assigned for a bad VTOC record.	3
Get Alternate Track	Replace defective record zero.	4

Figure 108. IEHATLAS Example Directory

Examples that use **disk** in place of actual device numbers must be changed before use. See "DASD and Tape Device Support" on page 3 for valid device number notation.

IEHATLAS EXAMPLE 1

In this example, the data set defined by SYSUT1 contains the bad record. An alternate track on the specified unit and volume is assigned to replace the defective track. Valid records from the defective track are copied to the alternate track and the replacement record (from SYSIN) is also written to the alternate track.

```
//JOBATLAS JOB  06#990,SMITH,MSGLEVEL=1
//STEP     EXEC PGM=IEHATLAS
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=NEWSET,UNIT=disk,VOLUME=SER=333333,
//              DISP=OLD
//SYSIN    DD   *
     TRACK=00000002000422020006S
           F3F1C2C2F0F00000
/*
```

The control statements are discussed below:

- SYSPRINT DD defines the device to which the output messages can be written (in this case, the system printer).
- SYSUT1 DD defines the data set (NEWSET) that contains the bad record.
- SYSIN DD defines the control data set, which follows in the input stream.
- TRACK specifies the cylinder and track number for the defective track, and the record number, key length, and data length of the bad record. In this example, the input record is to be placed on cylinder 2 (cccc=0002), track 4 (hhhh=0004), record 22 (rr=22); it has a key length of two (kk=02) with a logical record (data) length of six (dddd=0006). The WRITE special (S) character is used because there is a track overflow condition.

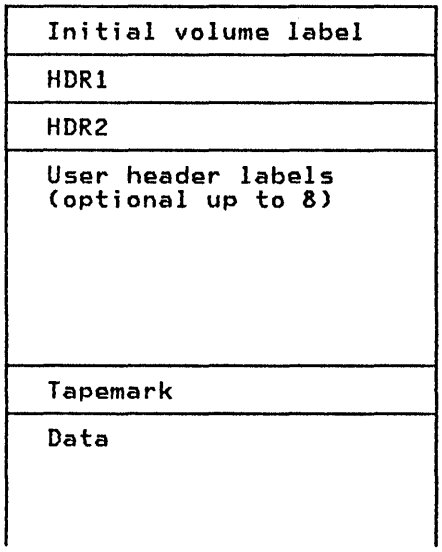


Figure 109. IBM Standard Label Group after Volume Receives Data

INPUT AND OUTPUT

IEHINITT uses as input a control data set that contains the utility control statements.

IEHINITT produces an output data set that contains:

- Utility program identification
- Initial volume label information for each successfully labeled tape volume.
- Contents of utility control statements.
- Any error messages.

RETURN CODES

IEHINITT returns a code in register 15 to indicate the results of program execution. The return codes and their meanings are listed below.

Codes	Meaning
00 (00 hex)	Successful completion. A message data set was created.
04 (04)	Successful completion. No message data set was defined by the user.
08 (08)	IEHINITT completed its operation, but error conditions were encountered during processing. A message data set was created.
12 (0C)	IEHINITT completed its operation, but error conditions were encountered during processing. No message data set was defined by the user.
16 (10)	IEHINITT terminated operation because of error conditions encountered while attempting to read the control data set. A message data set was created if defined by the user.

Figure 110. IEHINITT Return Codes

CONTROL

IEHINITT is controlled by job control statements and utility control statements. The job control statements are used to execute or invoke IEHINITT and to define data sets used and produced by IEHINITT. The utility control statement is used to specify applicable label information.

JOB CONTROL STATEMENTS

Figure 111 on page 337 shows the job control statements for IEHINITT.

PARM Information on the EXEC Statement

The EXEC statement can include PARM information that specifies the number of lines to be printed between headings in the message data set, as follows:

```
PARM='LINECNT=nn'
```

If PARM is omitted, 60 lines are printed between headings.

If IEHINITT is invoked, the line count option can be passed in a parameter list that is referred to by the optionaddr subparameter of the LINK or ATTACH macro instruction. In addition, a page count can be passed in a 6-byte parameter list that is referred to by the hdingaddr subparameter of the LINK or ATTACH macro instruction. For a discussion of linkage conventions, refer to "Invoking Utility Programs from a Problem Program" on page 13.

SYSPRINT DD Statement

The SYSPRINT data set must have a logical record length of 121 bytes. It must consist of fixed length records with an ISO/ANSI control character in the first byte of each record. Any blocking factor can be specified.

Statement	Use
JOB	Initiates the job.
EXEC	Specifies the program name (PGM=IEHINITT) or, if the job control statements reside in a procedure library, the procedure name. The EXEC statement can include additional PARM information; see "PARM Information on the EXEC Statement."
SYSPRINT DD	Defines a sequential output data set.
anyname DD	Defines a tape unit to be used in a labeling operation; more than one tape unit can be identified.
SYSIN DD	Defines the control data set. The control data set normally resides in the input stream; however, it can be defined as a member of a partitioned data set or as a sequential data set outside the input stream.

Figure 111. IEHINITT Job Control Statements

anyname DD Statement

The "anyname" DD statement is entered:

```
//anyname DD DCB=DEN=x,UNIT=(xxxx,n,DEFER)
```

The DEN parameter specifies the density at which the labels are written. The UNIT parameter specifies the device type, number of units to be used for the labeling operation, and deferred mounting. See the publication JCL for more information on the DEN and UNIT parameters.

The name "anyname" must be identical to a name specified in a utility control statement to relate the specified unit(s) to the utility control statement.

SYSIN DD Statement

The SYSIN data set must have a logical record length of 80. Any blocking factor can be specified.

UTILITY CONTROL STATEMENT

IEHINITT uses the utility control statement INITT to provide control information for a labeling operation.

Continuation requirements for utility control statements are described in "Continuing Utility Control Statements" on page 5.

INITT Statement

The INITT statement provides control information for the IEHINITT program.

Any number of INITT utility control statements can be included for a given execution of the program. An identically named DD statement must exist for a utility control statement in the job step.

Figure 112 shows a printout of a message data set including the INITT statement and initial volume label information. In this example, one INITT statement was used to place serial numbers 001122 and 001123 on two SL tape volumes. VOL1001122 and VOL1001123 are interpreted, as follows:

- VOL1 indicates that an initial volume label was successfully written to a tape volume.
- 001122 and 001123 are the serial numbers that were written onto the volumes.
- A blank space following the serial number represents the Volume Security field, which is not used during OPEN/CLOSE/EOV processing on an SL tape.

No errors occurred during processing.

```
SYSTEM SUPPORT UTILITIES   IEHINITT
                                                                  72
ALL   INITT  SER=001122,NUMBTAPE=2,OWNER='P.T.BROWN',              X
             DISP=REWIND

VOL1001122                      P.T.BROWN
VOL1001123                      P.T.BROWN
```

Figure 112. Printout of INITT Statement Specifications and Initial Volume Label Information

The format of the INITT statement is:

ddname	INITT	SER=xxxxxx ,DISP={REWIND|UNLOAD} [,OWNER='ccccccccccc[cccc]'] [,NUMBTAPE=n|1] [,LABTYPE=AL] [,ACCESS=c]

Parameters	Applicable Control Statements	Description of Parameters
ACCESS	INITT	**ACCESS=**c specifies the ISO/ANSI volume accessibility code. Valid values for c are upper case A through Z only. The default value is a blank character, indicating unlimited access to the volume. You **cannot** specify a blank character for the access code. The Volume Access installation exit routine in MVS must be modified to allow subsequent use of the volume if ACCESS is specified. Refer to Magnetic Tape Labels and File Structure for further information about volume accessibility and ISO/ANSI installation exits. ACCESS is invalid unless LABTYPE=AL has also been specified.
DISP	INITT	**DISP={REWIND|UNLOAD}** specifies whether a tape is to be rewound or rewound and unloaded. These values can be coded: **REWIND** specifies that a tape is to be rewound (but not unloaded) after the label has been written. **UNLOAD** specifies that a tape is to be rewound and unloaded after the label has been written. This is the default.
LABTYPE	INITT	**LABTYPE=AL** specifies that an ISCII/ASCII volume label written in ISO/ANSI Version 3 format is to be created. Labels written in ISO/ANSI cannot be put on a 7-track tape volume. **Default:** The tape is written in EBCDIC for 9-track tape volumes and in BCD for 7-track tape volumes.
ddname	INITT	ddname specifies the name that is identical to the ddname in the name field of the DD statement defining a tape unit(s). This name must begin in column 1.
NUMBTAPE	INITT	**NUMBTAPE=**n|1 specifies the number of tapes to be labeled according to the specifications made in this control statement. The value n represents a number from 1 to 255. If more than one tape is specified, the volume serial number of the first tape must be numeric.

Parameters	Applicable Control Statements	Description of Parameters
OWNER	INITT	**OWNER='**cccccccccc[cccc]**'** specifies the owner's name or similar identification. The information is specified as character constants, and can be up to 10 bytes in length for EBCDIC and BCD volume labels, or up to 14 bytes in length for volume labels written in ISCII/ASCII. The delimiting apostrophes must be present if blanks, commas, apostrophes, equal signs, or other special characters (except periods or hyphens) are included. The set of valid ISO/ANSI 'a' type characters for ISCII/ASCII tapes is as follows: upper case A-Z, numeric 0-9, and special characters **!"%&'()*+,-./:;<=>?** If an apostrophe is included within the OWNER name field, it must be written as two consecutive apostrophes.
SER	INITT	**SER=**xxxxxx specifies the volume serial number of the first or only tape to be labeled. For IBM standard labeled (SL) tapes, the serial number cannot contain blanks, commas, apostrophes, equal signs, or special characters other than periods or hyphens. ISO/ANSI labeled tapes (AL) may contain any valid ISO/ANSI 'a' type character as described under the OWNER keyword. However, if any nonalphameric character (including a period or a hyphen) is present, delimiting apostrophes must be included. You cannot use a blank as the first character in a volume serial number. A specified serial number is increased by one for each additional tape to be labeled. (Serial number 999999 is incremented to 000000.) When processing multiple tapes, the volume serial number must be all numeric.

IEHINITT EXAMPLES

The following examples illustrate some of the uses of IEHINITT. Figure 113 can be used as a quick-reference guide to IEHINITT examples. The numbers in the "Example" column refer to examples that follow.

Operation	Comments	Example
LABEL	Three 9-track tapes are to be labeled.	1

Figure 113 (Part 1 of 2). IEHINITT Example Directory

Operation	Comments	Example
LABEL	A 9-track tape is to be labeled.	2
LABEL	Two groups of 9-track tape volumes are to be labeled.	3
LABEL	9-track tape volumes are to be labeled. Sequence numbers are to be incremented by 10.	4
LABEL	Three 9-track tape volumes are to be labeled. An alphameric label is to be placed on a tape volume; numeric labels are placed on the remaining two tape volumes.	5
LABEL	Two 9-track tape volumes are to be labeled. The first volume is labeled at a density of 6250 bpi; the second at a density of 1600 bpi.	6
LABEL	A 9-track tape volume is labeled in ISO/ANSI format with a nonblank access code.	7

Figure 113 (Part 2 of 2). IEHINITT Example Directory

Examples that use **tape** in place of actual device numbers must be changed before use. See "DASD and Tape Device Support" on page 3 for valid device number notation.

IEHINITT EXAMPLE 1

In this example, serial numbers 001234, 001235, and 001236 are placed on three tape volumes; the labels are written in EBCDIC at 800 bits per inch. Each volume labeled is mounted, when it is required, on a single 9-track tape unit.

```
//LABEL1   JOB  09#990,BROWN,MSGLEVEL=(1,1)
//         EXEC PGM=IEHINITT
//SYSPRINT DD   SYSOUT=A
//LABEL    DD   DCB=DEN=2,UNIT=(tape,1,DEFER)
//SYSIN    DD   *
LABEL   INITT   SER=001234,NUMBTAPE=3
/*
```

The control statements are discussed below:

- LABEL DD defines the tape unit used in the labeling operation.
- SYSIN DD defines the control data set, which follows in the input stream.
- LABEL INITT specifies the number of tapes to be labeled (3), beginning with 001234.

IEHINITT EXAMPLE 2

In this example, serial number 001001 is placed on one ISO/ANSI tape volume; the label is written at 800 bits per inch. The volume labeled is mounted, when it is required, on a 9-track tape unit.

```
//LABEL2    JOB  09#990,BROWN,MSGLEVEL=(1,1)
//          EXEC PGM=IEHINITT
//SYSPRINT  DD   SYSOUT=A
//ASCIILAB  DD   DCB=DEN=2,UNIT=(tape,1,DEFER)
//SYSIN     DD   *
ASCIILAB INITT   SER=001001,OWNER='SAM A. BROWN',LABTYPE=AL
/*
```

The control statements are discussed below:

- ASCIILAB DD defines the tape volume to be used in the labeling operation.
- SYSIN DD defines the control data set, which follows in the input stream.
- ASCIILAB INITT specifies the serial number, owner ID and label type for the volume.

IEHINITT EXAMPLE 3

In this example, two groups of serial numbers (001234, 001235, 001236, and 001334, 001335, 001336) are placed on six tape volumes. The labels are written in EBCDIC at 800 bits per inch. Each volume labeled is mounted, when it is required, on a single 9-track tape unit.

```
//LABEL3   JOB  09#990,BROWN,MSGLEVEL=(1,1)
//         EXEC PGM=IEHINITT
//SYSPRINT DD   SYSOUT=A
//LABEL    DD   DCB=DEN=2,UNIT=(tape,1,DEFER)
//SYSIN    DD   *
LABEL   INITT   SER=001234,NUMBTAPE=3
LABEL   INITT   SER=001334,NUMBTAPE=3
/*
```

The control statements are discussed below:

- LABEL DD defines the tape unit to be used in the labeling operation.
- SYSIN DD defines the control data set, which follows in the input stream.
- LABEL INITT defines the two groups of serial numbers to be put on six tape volumes.

IEHINITT EXAMPLE 4

In this example, serial numbers 001234, 001244, 001254, 001264, 001274, etc., are placed on eight tape volumes. The labels are written in EBCDIC at 800 bits per inch. Each volume labeled is mounted, when it is required, on one of four 9-track tape units.

```
//LABEL4   JOB  09#990,BROWN,MSGLEVEL=(1,1)
//        EXEC PGM=IEHINITT
//SYSPRINT DD   SYSOUT=A
//LABEL    DD   DCB=DEN=2,UNIT=(tape,4,DEFER)
//SYSIN    DD   *
LABEL   INITT   SER=001234
LABEL   INITT   SER=001244
LABEL   INITT   SER=001254
LABEL   INITT   SER=001264
LABEL   INITT   SER=001274
LABEL   INITT   SER=001284
LABEL   INITT   SER=001294
LABEL   INITT   SER=001304
/*
```

The control statements are discussed below:

- LABEL DD defines the tape unit used in the labeling operation.
- SYSIN DD defines the control data set, which follows in the input stream.
- The LABEL INITT statements define the tapes to be labeled by volume serial number.

IEHINITT EXAMPLE 5

In this example, serial number TAPE1 is placed on a tape volume, and serial numbers 001234 and 001235 are placed on two tape volumes. The labels are written in EBCDIC at 800 and 1600 bits per inch, respectively.

```
//LABEL5   JOB  09#990,BROWN,MSGLEVEL=(1,1)
//        EXEC PGM=IEHINITT
//SYSPRINT DD   SYSOUT=A
//LABEL1   DD   DCB=DEN=2,UNIT=(tape,1,DEFER)
//LABEL2   DD   DCB=DEN=3,UNIT=(tape,1,DEFER)
//SYSIN    DD   *
LABEL1  INITT   SER=TAPE1
LABEL2  INITT   SER=001234,NUMBTAPE=2
/*
```

The control statements are discussed below:

- LABEL1 DD and LABEL2 DD define two tape volumes to be used in the labeling operation.
- SYSIN DD defines the control data set, which follows in the input stream.
- LABEL1 INITT places the serial number TAPE1 on the tape volume defined in LABEL1 DD. LABEL2 INITT places the serial numbers 001234 and 001235 on the tape volume defined in LABEL2 DD.

IEHINITT EXAMPLE 6

In this example, the serial number 006250 is written in EBCDIC on a tape volume at a density of 6250 bpi, and the serial number 001600 is written in EBCDIC on a second volume at a density of 1600 bpi.

```
//LABEL6   JOB  09#990,BROWN,MSGLEVEL=(1,1)
//        EXEC PGM=IEHINITT
//SYSPRINT DD   SYSOUT=A
//DDFIRST  DD   DCB=DEN=4,UNIT=(tape,1,DEFER)
//DDSECOND DD   DCB=DEN=3,UNIT=(tape,1,DEFER)
//SYSIN    DD   *
DDFIRST  INITT  SER=006250
DDSECOND INITT  SER=001600
/*
```

The control statements are discussed below:

- DDFIRST DD defines the first tape volume to be used.
- DDSECOND DD defines the second tape volume to be used.
- SYSIN DD defines the control data set, which follows in the input stream.
- DDFIRST INITT writes the serial number 006250 on the volume defined in DDFIRST DD. DDSECOND INITT writes the serial number 001600 on the volume defined in DDSECOND DD.

IEHINITT EXAMPLE 7

In this example, an ISO/ANSI (AL) labeled tape is created with a nonblank access code. The volume serial number is TAPE01.

```
//LABEL7   JOB  09#990,BLUE,MSGLEVEL=(1,1)
//STEP01   EXEC PGM=IEHINITT
//SYSPRINT DD   SYSOUT=A
//LABEL    DD   UNIT=(tape,1,DEFER),DCB=DEN=4
//SYSIN    DD   *
LABEL  INITT  SER=TAPE01,OWNER=TAPOWNER,LABTYPE=AL,ACCESS=A
/*
```

The control statements are discussed below.

- LABEL DD defines the device on which the tape is mounted.
- SYSIN DD defines the control data set, which follows in the input stream.
- The INITT statement creates an ISO/ANSI label for the tape with volume serial number TAPE01, owned by TAPOWNER. The ACCESS code is specified as "A", and the MVS operating system which receives this volume must be able to recognize the "A" in order for the volume to be accepted.

IEHLIST is a system utility used to list entries in an OS CVOL, entries in the directory of one or more partitioned data sets, or entries in an indexed or non-indexed volume table of contents. Any number of listings can be requested in a single execution of the program.

LISTING OS CVOL ENTRIES

IEHLIST lists all OS CVOL entries that are part of the structure of a fully qualified, data set name. Figure 114 shows an index structure for which IEHLIST lists fully qualified names A.B.D.W, A.B.D.X, A.B.E.Y, and A.B.E.Z. Because A.C.F does not represent a cataloged data set (that is, the lowest level of qualification has been deleted), it is not a fully qualified name, and it is not listed.

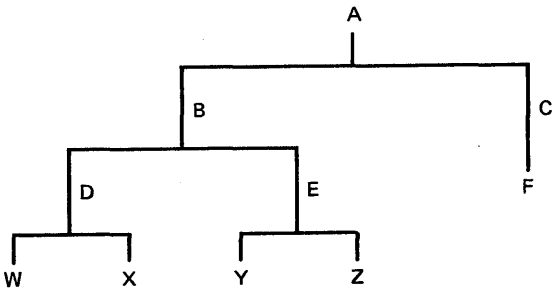


Figure 114. Index Structure—Listed by IEHLIST

IEHLIST will list only OS CVOLs (SYSCTLG data sets). To list ICF or VSAM catalogs, use access method services. See Access Method Services Reference for more information.

LISTING A PARTITIONED DATA SET DIRECTORY

IEHLIST can list up to ten partitioned data set directories at a time. A partitioned directory is composed of variable length records blocked into 256-byte blocks. Each directory block can contain one or more entries which reflect member (and/or alias) names and other attributes of the partitioned members. IEHLIST can list these blocks in edited and unedited format.

Figure 115 on page 346 shows a directory block as it exists in storage.

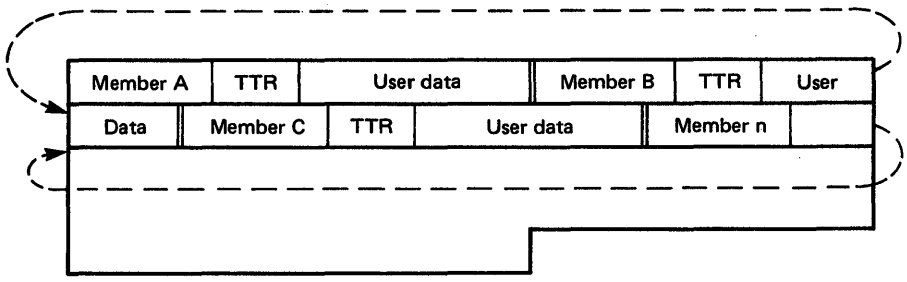


Figure 115. Sample Directory Block

Edited Format

IEHLIST optionally provides the following information, which is obtained from the applicable partitioned data set directory, when an edited format is requested:

- Member name
- Entry point
- Relative address of start of member
- Relative address of start of text
- Contiguous virtual storage requirements
- Length of first block of text
- Origin of first block of text
- System status indicators
- Linkage editor attributes
- APF authorization required
- Other information

Figure 116 shows an edited entry for a partitioned member (IEANUC01). The entry is shown as it is listed by the IEHLIST program.

```
                                        OTHER INFORMATION INDEX
    SCATTER FORMAT  SCTR=SCATTER/TRANSLATION TABLE TTR IN HEX, LEN OF SCTR LIST IN DEC, LEN OF TRANS TABLE IN DEC,
                          ESDID OF FIRST TEXT RCD IN DEC, ESDID OF CSECT CONTAINING ENTRY POINT IN DEC
    OVERLAY FORMAT  ONLY=NOTE LIST RCD TTR IN HEX, NUMBER OF ENTRIES IN NOTE LIST RCD IN DEC
    ALIAS NAMES     ALIAS MEMBER NAMES WILL BE FOLLOWED BY AN ASTERISK IN THE PDS FORMAT LISTING
                                              ATTRIBUTE INDEX
    BIT   ON      OFF         BIT   ON      OFF          BIT   ON          OFF         BIT   ON         OFF
     0    RENT    NOT RENT     4    OL      NOT OL       8    NOT DC     DC           12    NOT EDIT   EDIT
     1    REUS    NOT REUS     5    SCTR    BLOCK        9    ZERO ORG   NOT ZERO     13    SYMS       NO SYMS
     2    ONLY    NOT ONLY     6    EXEC    NOT EXEC     10   EP ZERO    NOT ZERO     14    F LEVEL    E LEVEL
     3    TEST    NOT TEST     7    1 TXT   MULTI RCD    11   NO RLD     RLD          15    REFR       NOT REFR
MEMBER     ENTRY     ATTR   REL      ADDR-HEX   CONTIG      LEN 1ST   ORG 1ST   SST       VS       AUTH OTHER
NAME       PT-HEX    HEX    BEGIN    1ST TXT    STOR-DEC    TXT-DEC   TXT-HEX   INFO      ATTR     REQ  INFORMATION
IEANUC01   000000    06E2   000004   00020F     000166248   0927                ABSENT    880000   NO   SCTR=000000,
                                                                                                        00484,01084,32,32
OF THE 00002 DIRECTORY BLOCKS ALLOCATED TO THIS PDS, 00001 ARE(IS) COMPLETELY UNUSED
```

Figure 116. Edited Partitioned Directory Entry

Before printing the directory entries on the first page, an index is printed explaining the asterisk (*), if any, following a member name, the **attributes** (fields 3 and 10), and **other information** (field 12). Under OTHER INFORMATION INDEX, scatter and overlay format data is described positionally as it appears in the listing; under the ATTRIBUTE INDEX, the meaning of each attribute bit is explained.

Each directory entry occupies one printed line, except when the member name is an alias and the main member name and associated entry point appear in the user data field. When this occurs, two lines are used and every alias is followed by an asterisk. If the main member is renamed, the old member name will still be in the alias directory entry and consequently printed on the second line.

The FORMAT option of the LISTPDS statement applies only to a partitioned data set whose members have been created by the linkage editor (that is, the directory entries are at least 34 bytes long). If a directory entry is less than 34 bytes, a message is issued and the entry is printed in unedited format; if the entry is longer than 34 bytes, it is assumed that it is created by the linkage editor.

Unedited (Dump) Format

The user may choose the unedited format. If this is the case, IEHLIST lists each member separately.

Figure 117 shows how the information in Figure 115 on page 346 is listed.

Note: A listing organized as shown in Figure 117 can also be obtained by using IEBPTPCH (see "IEBPTPCH Program" on page 241).

```
MEMB A      TTR          USER DATA

MEMB B      TTR          USER DATA

MEMB C      TTR          USER DATA

MEMB n      TTR          USER DATA
```

Figure 117. Sample Partitioned Directory Listing

To correctly interpret user data information, the user must know the format of the partitioned entry. The formats of directory entries are discussed in the Debugging Handbook.

LISTING A VOLUME TABLE OF CONTENTS

IEHLIST can be used to list, partially or completely, entries in a specified volume table of contents (VTOC), whether indexed or nonindexed. The program lists the contents of selected data set control blocks (DSCBs) in edited or unedited form.

For more information on indexed VTOCs, including a description of the VPSM, VIXM and VMDS, see System Programming Library: Data Management.

Edited Format

Two edited formats are available.

FIRST EDITED FORMAT: The first edited format is a comprehensive listing of the DSCBs in the VTOC. It provides the status and attributes of the volume, and describes in depth the data sets residing on the volume. This listing includes:

- Logical record length and block size
- Initial and secondary allocations
- Upper and lower limits of extents
- Alternate track information
- Available space information, including totals of unallocated cylinders, unallocated tracks, and unallocated (Format 0) DSCBs
- Option codes (printed as two hexadecimal digits)
- Record formats

A VTOC consists of as many as seven types of DSCBs which contain information about the data sets residing on the volume:

- Identifier DSCB—Format 1
- Index DSCB—Format 2
- Extension DSCB—Format 3
- VTOC DSCB—Format 4
- Free Space DSCB—Format 5
- Shared Extent DSCB—Format 6
- Free VTOC DSCB—Format 0

The first DSCB in a VTOC (and on your listing) is always a VTOC (Format 4) DSCB. It defines the scope of the VTOC itself; that is, it contains information about the VTOC and the volume rather than the data sets referenced by the VTOC.

The DSCB is followed by the Free Space (Format 5) DSCB, which describes the space available on the volume for allocation to other data sets. More than one Format 5 DSCB may be required to describe the available space on a volume because each Format 5 DSCB describes up to 26 extents.

The Format 4 and Format 5 DSCBs are followed, in any order, by Format 1, 2, 3, or 6 DSCBs.

Each Identifier (Format 1) DSCB contains information about a particular data set or VSAM data space residing on the volume. This type of DSCB describes the characteristics and up to three extents of the data set.

For data sets having indexed sequential organization, additional characteristics are specified in an Index (Format 2) DSCB pointed to by the Identifier (Format 1) DSCB.

Additional extents are described in an Extension (Format 3) DSCB pointed to by the Identifier (Format 1) DSCB or in the Index (Format 2) DSCB for an ISAM data set.

A Shared Extent (Format 6) DSCB is used for shared-cylinder allocation. It describes the extent of space (one or more contiguous cylinders) that is being shared by two or more data sets. The Shared Extent (Format 6) DSCB is pointed to by the VTOC (Format 4) DSCB. Subsequent Format 6 DSCBs are pointed to by the previous Format 6 DSCB. Though shared extent data sets cannot be created by MVS/370, they are supported if previously created.

A Free VTOC Record (Format 0) DSCB, which indicates space available for another DSCB, is not listed by IEHLIST. They are 140-byte records, consisting of binary zeros, that are overwritten with Format 1, 2, or 3 DSCBs when a new data set is allocated, with Format 5 DSCBs when space is released, or with Format 3 DSCBs when a Format 1 or Format 2 must be extended.

Indexed VTOCs: For indexed VTOCs, there are two types of formatted listings. These types are specified using the INDEXDSN parameter.

If INDEXDSN is omitted, the listing contains:

- A statement of the number of levels in the index, if enabled.
- A formatted Format 4 DSCB.
- Formatted data set entries in alphameric order (Format 1 DSCB physical-sequential order if the index is disabled).
- Formatted VPSM freespace information.
- Totals of unallocated cylinders, unallocated tracks, unallocated (Format 0) DSCBs, and unallocated VIRs.

If INDEXDSN=name is specified, the listing contains, in addition to the items above:

- A formatted VPSM, VMDS, and VIXM.
- Allocated VIERs, formatted and listed by level and key sequence within level (in physical-sequential order if the index is disabled).
- If the VTOC index is disabled, a statement is included to this effect.

Figure 118 on page 350 shows a sample listing of the first edited format. This sample illustrates how each DSCB will appear on a listing, although in many cases the VTOC may not contain all possible types. The information is in columns, with the values or numbers appearing underneath each item's heading.

SECOND EDITED FORMAT: The second edited format is an abbreviated description of the data sets. It is provided by default when no format is requested specifically. It provides the following information:

- Data set name
- Creation date (dddyy)
- Expiration date (dddyy)
- Password indication
- Organization of the data set
- Extent(s)
- Volume serial number

The last line in the listing indicates how much space remains in the VTOC.

For nonindexed VTOCs, data set entries are listed in physical-sequential order. Totals of unallocated cylinders, unallocated tracks, and unallocated (Format 0) DSCBs are also listed.

```
                        SYSTEMS SUPPORT UTILITIES---IEHLIST                              PAGE 1

              CONTENTS OF VTOC ON VOL EXAMPL

FORMAT 4 DSCB NO AVAIL/MAX DSCB /MAX DIRECT  NO AVAIL NEXT ALT     FORMAT 6    LAST FMT 1      VTOC EXTENT       THIS DSCB
          VI   DSCBS    PER TRK  BLK PER TRK  ALT TRK  TRK(C-H)    (C-H-R)     DSCB(C-H-R)/LOW(C-H) HIGH(C-H)   (C-H-R)
          00     154        16          10         30    200                      5   0   5    5   0   5   9    5   0   1

FORMAT 5 DSCB      A = NUMBER OF TRKS IN ADDITION TO FULL CYLS IN THE EXTENT
      TRK  FULL          TRK  FULL          TRK  FULL          TRK  FULL          TRK  FULL          TRK  FULL
     ADDR  CYLS   A     ADDR  CYLS   A     ADDR  CYLS   A     ADDR  CYLS   A     ADDR  CYLS   A     ADDR  CYLS   A
       17     3   3      110   189   0

               DSCB(C-H-R)   5   0   2

--------------DATA SET NAME---------------  ID  SER NO  SEQ NO  CREDT  EXPDT  NO EXT  DSORG  RECFM  OPTCD  BLKSIZE
EXAMPLE.OF.COMBINED.FORMATS.ONE.AND.TWO     1  EXAMPL       1  36699  27469      1     IS     F                100

            LRECL  KEYLEN  INITIAL ALLOC  2ND ALLOC/LAST BLK PTR(T-R-L)  USED PDS BYTES  FMT 2 OR 3(C-H-R)/DSCB(C-H-R)
              100      4      ABSTR                0                                        5   0   3     5   0   4

            EXTENTS  NO  LOW(C-H)   HIGH(C-H)
                     0      6   0       10   9

     2MIND(M-B-C-H)/3MIND(M-B-C-H)/L2MFN(C-H-R)/L3MIN(C-H-R)/CYLAD(M-B-C-H)/ADLIN(M-B-C-H)/ADHIN(M-B-C-H)/NOBYT/ NOTRK
       0 0   0   0    0 0   0   0    0   0   0    0   0   0    1 0  10   9    0 0   0   0    1 0  10   9     70     0

     LTRAD(C-H-R)/LCYAD(C-H-R)/LMSAD(C-H-R)/LPRAD(M-B-C-H-R) /NOLFV /CYLOV/ TAGDT/   PRCTR  / OVRCT/ RORG1/PTRDS(C-H-R)
       6   0   3   10   9   1    0   0   0    1 0   6   1  12    1       0            20          0      0
                              ----UNABLE TO CALCULATE EMPTY SPACE.

--------------DATA SET NAME---------------  ID  SER NO  SEQ NO  CREDT  EXPDT  NO EXT  DSORG  RECFM  OPTCD  BLKSIZE
EXAMPLE.OF.COMBINED.FORMATS.ONE.AND.THREE   1  EXAMPL       1  36699  27069     16     PS     V                3504

            LRECL  KEYLEN  INITIAL ALLOC  2ND ALLOC/LAST BLK PTR(T-R-L)  USED PDS BYTES  FMT 2 OR 3(C-H-R)/DSCB(C-H-R)
             3500          TRKS                 1          15   1  1723                     5   0   6     5   0   5

            EXTENTS  NO  LOW(C-H)  HIGH(C-H)   NO  LOW(C-H)  HIGH(C-H)    NO  LOW(C-H)  HIGH(C-H)
                     0     0   1      0   1     1     0   2      0   2      2     0   3      0   3
                     3     0   4      0   4     4     0   5      0   5      5     0   6      0   6
                     6     0   7      0   7     7     0   8      0   8      8     0   9      0   9
                     9     1   0      1   0    10     1   1      1   1     11     1   2      1   2
                    12     1   3      1   3    13     1   4      1   4     14     1   5      1   5
                    15     1   6      1   6
                              ----ON THE ABOVE DATA SET,THERE ARE  0 EMPTY TRACK(S).

THERE ARE  192 EMPTY CYLINDERS PLUS    3 EMPTY TRACKS ON THIS VOLUME
THERE ARE  154 BLANK DSCBS IN THE VTOC ON THIS VOLUME
```

Figure 118. Sample Printout of a Volume Table of Contents

For indexed VTOCs, this listing contains:

- A statement of the number of levels in the index.
- Data set entries listed in alphameric order.
- Totals of unallocated cylinders, unallocated tracks, unallocated (Format 0) DSCBs, and unallocated VIRs.

Unedited (Dump) Format

This option produces a complete hexadecimal listing of the DSCBs in the VTOC. The listing is in an unedited dump form, requiring the user to know the various formats of applicable DSCBs. The VTOC overlay for IEHLIST listings of VTOCs in dump format (form number 620-0109-1) is useful in identifying the fields of the DSCBs.

For nonindexed VTOCs, this listing contains:

- DSCBs dumped in physical-sequential order.
- Totals of unallocated cylinders, unallocated tracks, and unallocated (Format 0) DSCBs.

For indexed VTOCs there are two types of dump listings. These types are specified using the INDEXDSN parameter.

If INDEXDSN is omitted, the listing contains:

- DSCBs dumped in physical-sequential order (one token Format 5 DSCB is identified).
- A dump of the VPSM.
- Totals of unallocated cylinders, unallocated tracks, unallocated (Format 0) DSCBs, and unallocated VIRs.

If INDEXDSN=name is specified, the listing contains, in addition to the items above:

- Dumps of the VIXM and the VMDS.
- A dump of all allocated VIERs dumped in hierarchic order. All VIERs at the highest level are dumped, starting with the VIER with the lowest high key; next, all VIERs at the next lower level are dumped, starting with the VIER with the lowest high key. The listing continues in this manner until all VIERs at level 1 are dumped.

  If the VTOC index is disabled, both allocated and unallocated VIERs are dumped in physical-sequential order.
- If the VTOC index is disabled, a statement is included to this effect.

Refer to Debugging Handbook for a discussion of the various formats that data set control blocks can assume.

INPUT AND OUTPUT

IEHLIST uses the following input:

- One or more source data sets that contain the data to be listed. The input data set(s) can be:

  1. A VTOC,
  2. A partitioned data set,
  3. An OS CVOL (SYSCTLG).
- A control data set, which contains utility control statements that are used to control the functions of IEHLIST.

IEHLIST produces as output a message data set which contains the result of the IEHLIST operations. The message data set includes the listed data and any error messages.

RETURN CODES

IEHLIST returns a code in register 15 to indicate the results of program execution. The return codes and their meanings are listed below.

Codes	Meaning
00 (00 hex)	Successful completion.
08 (08)	An error condition caused a specified request to be ignored. Processing continues.
12 (0C)	A permanent input/output error occurred. The job is terminated.
16 (10)	An unrecoverable error occurred while reading the data set. The job is terminated.

Figure 119. IEHLIST Return Codes

CONTROL

IEHLIST is controlled by job control statements and utility control statements. The job control statements are used to execute or invoke IEHLIST and to define the data sets used and produced by IEHLIST.

Utility control statements are used to control the functions of the program and to define those data sets or volumes to be modified.

JOB CONTROL STATEMENTS

Figure 120 on page 353 shows the job control statements for IEHLIST.

With the exception of the SYSIN and SYSPRINT DD statements, all DD statements in this table are used as device allocation statements, rather than as true data definition statements.

Concatenated DD statements are allowed only for SYSIN.

Because IEHLIST modifies the internal control blocks created by device allocation DD statements, IEHLIST job control statements must not include the DSNAME parameter. (All data sets are defined explicitly or implicitly by utility control statements.)

IEHLIST cannot support empty space calculations for OS CVOL data sets allocated in blocks when the block sizes are approximately the same or larger than the track size. The empty block calculation gives only approximate indications of available space. When IEHLIST cannot supply an approximate number, the "Unable to Calculate" message is issued.

IEHLIST specifications do not allow for protection of the object being listed. If another program updates a block of the data set just prior to IEHLIST reading the data set, a message (IEH105I or IEH108I) may be issued and the output produced by IEHLIST may be incorrect. If this happens, rerun the job.

PARM Information on the EXEC Statement

Additional information can be specified in the PARM parameter of the EXEC statement to control the number of lines printed per page. The PARM parameter can be coded:

```
PARM='LINECNT=xx'
```

Statement	Use
JOB	Initiates the job.
EXEC	Specifies the program name (PGM=IEHLIST) or, if the job control statements reside in a procedure library, the procedure name. Additional PARM information can be specified to control the number of lines printed per page. See "PARM Information on the EXEC Statement."
SYSPRINT DD	Defines a sequential message data set.
anyname1 DD	Defines a permanently mounted volume.
anyname2 DD	Defines a mountable device type. This can be specified more than once as long as each "anyname" is unique.
SYSIN DD	Defines the control data set. The control data set normally follows the job control language in the input stream; however, it can be defined as an unblocked sequential data set or member of a procedure library.

Figure 120. IEHLIST Job Control Statements

The LINECNT parameter specifies the number of lines, xx, to be printed per page; xx is a decimal number from 01 through 99. If LINECNT is not specified, 58 lines are printed per page. The PARM field cannot contain embedded blanks, zeros, or any other PARM keywords if LINECNT is specified.

SYSPRINT DD Statement

The block size for SYSPRINT must be a multiple of 121. Any blocking factor can be specified for this block size.

anyname1 DD Statement

An "anyname1" DD statement must be included for each permanently mounted volume referred to in the job step. (The system residence volume is considered to be a permanently mounted volume.)

The "anyname1" DD statement can be entered:

```
//anyname1 DD UNIT=xxxx,VOLUME=SER=xxxxxx,DISP=OLD
```

The UNIT and VOLUME=SER parameters define the device type and volume serial number. The DISP=OLD specification prevents the inadvertent deletion of the data set. (This statement has arbitrarily been assigned the ddname DD1 in the IEHLIST examples.)

anyname2 DD Statement

An "anyname2" DD statement must be included for each mountable device to be used in the job step.

When deferred mounting is required, the "anyname2" DD statement can be entered:

```
//anyname2 DD UNIT=(xxxx,,DEFER),VOLUME=(PRIVATE,...),DISP=OLD
```

(This statement is arbitrarily assigned the ddname DD2 in the IEHLIST examples.)

When IEHLIST is dynamically invoked in a job step by another program, the DD statements defining mountable devices for IEHLIST must precede DD statements required by the other program.

Unit affinity cannot be used on DD statements defining mountable devices.

See Appendix B, "DD Statements for Defining Mountable Devices" on page 443 for information on defining mountable devices.

SYSIN DD Statement

The block size for SYSIN must be a multiple of 80. Any blocking factor can be specified for this block size.

UTILITY CONTROL STATEMENTS

Figure 121 shows the utility control statements for IEHLIST.

Statement	Use
LISTCTLG	Requests a listing of all or part of an OS CVOL (SYSCTLG).
LISTPDS	Requests a directory listing of one or more partitioned data sets.
LISTVTOC	Requests a listing of all or part of a volume table of contents.

Figure 121. IEHLIST Utility Control Statements

Continuation requirements for utility control statements are described in "Continuing Utility Control Statements" on page 5.

LISTCTLG Statement

The LISTCTLG statement is used to request a listing of either the entire OS CVOL or a specified portion of the OS CVOL (SYSCTLG data set). The listing includes the fully qualified name of each applicable cataloged data set and the serial number of the volume on which it resides. **Empty index levels are not listed.**

The format of the LISTCTLG statement is:

[label]	LISTCTLG	[VOL=device=serial] [,NODE=name]

LISTPDS Statement

The LISTPDS statement is used to request a directory listing of one or more partitioned data sets that reside on the same volume.

Before printing the directory entries on the first page, an index is printed explaining the **attributes** (fields 3 and 10) and **other information** (field 12). OTHER INFORMATION INDEX explains scatter and overlay format data as it appears in the listing; ATTRIBUTE INDEX explains each attribute bit.

The FORMAT option of the LISTPDS statement may be used only on a partitioned data set whose members have been created by the linkage editor. Members that have not been created by the linkage editor cause their directory entries to be listed in unedited (DUMP) format.

The format of the LISTPDS statement is:

[label]	**LISTPDS**	**DSNAME=(**name**[,**name**]...)** **[,VOL=**device**=**serial**]** **[,DUMP|FORMAT]**

LISTVTOC Statement

The LISTVTOC statement is used to request a partial or complete listing of the entries in a specified volume table of contents.

If you are using IEHLIST to list both the VTOC and the index data set of an indexed VTOC, refer to "Listing a Volume Table of Contents" on page 347.

The format of the LISTVTOC statement is:

[label]	**LISTVTOC**	**[DUMP|FORMAT]** **[,INDEXDSN=SYS1.VTOCIX.**xxxx**]** **[,DATE=**dddyy**]** **[,VOL=**device**=**serial**]** **[,DSNAME=(**name**[,**name**]...)]**

Parameters	Applicable Control Statements	Description of Parameters
DATE	LISTVTOC	**DATE=**dddyy specifies that each entry that expires before this date is to be flagged with an asterisk (*) after the entry name in the listing. This parameter applies only to the abbreviated edited format. The date is represented by ddd, the day of the year, and yy, the last two digits of the year. **Default:** No asterisks appear in the listing.

Parameters	Applicable Control Statements	Description of Parameters
DSNAME	LISTPDS	**DSNAME=(**name[,name]...**)** specifies the fully qualified names of the partitioned data sets whose directories are to be listed. A maximum of 10 names is allowed. If the list consists of only a single name, the parentheses can be omitted.
	LISTVTOC	**DSNAME=(**name[,name]...**)** specifies the fully qualified names of the data sets whose entries are to be listed. A maximum of 10 names is allowed. If the list consists of only a single name, the parentheses can be omitted.
DUMP	LISTPDS LISTVTOC	**DUMP** specifies that the listing is to be in unedited, hexadecimal form. **Default:** If both DUMP and FORMAT are omitted, an abbreviated edited format is generated for LISTVTOC. For LISTPDS, DUMP is the default used.
FORMAT	LISTPDS	**FORMAT** specifies that the listing is to be edited for each directory entry. The FORMAT option of the LISTPDS statement may be used only on a partitioned data set whose members have been created by the linkage editor. Members that have not been created by the linkage editor cause their directory entries to be listed in unedited (DUMP) format.
	LISTVTOC	**FORMAT** specifies that a comprehensive edited listing is to be generated. **Default:** If both FORMAT and DUMP are omitted, an abbreviated edited format is generated for LISTVTOC. For LISTPDS, DUMP is the default used.
INDEXDSN	LISTVTOC	**INDEXDSN=SYS1.VTOCIX.**xxxx specifies that index information is to be listed, in addition to the VTOC. xxxx is any third level qualifier. DUMP or FORMAT must be specified if INDEXDSN is specified. For more information on indexed VTOCs, refer to "Listing a Volume Table of Contents" on page 347.

Parameters	Applicable Control Statements	Description of Parameters
NODE	LISTCTLG	**NODE=**name specifies a qualified name. All data set entries whose names are qualified by this name are listed. The OS CVOL must be defined in the ICF or VSAM master catalog as: SYSTCTLG.VYYYYYY, where YYYYYY is the serial number of the OS CVOL. See Catalog Users Guide for details. **Default:** All data set entries are listed.
VOL	LISTCTLG LISTPDS LISTVTOC	**VOL=**device=serial specifies the device type and volume serial number of the volume on which the OS CVOL, PDS directory, or VTOC resides. For LISTPDS, if the partitioned data set is not on the system residence volume, the VOL parameter is required. **Default:** For LISTCTLG, the OS CVOL is assumed to reside on the system residence volume.

IEHLIST EXAMPLES

The following examples illustrate some of the uses of IEHLIST. Figure 122 can be used as a quick-reference guide to IEHLIST examples. The numbers in the "Example" column refer to examples that follow.

Operation	Devices	Comments	Example
LISTCTLG	Disk and system output device	Source OS CVOL is to be listed on the system output device.	1
LISTCTLG	Disk system residence device and system output device	Three OS CVOLs and part of a fourth are to be listed on the system output device.	2
LISTPDS	Disk and system output device	Three partitioned directories are to be listed on the system output device.	3
LISTVTOC	Disk and system output device	Volume table of contents is to be listed in edited form; selected data set control blocks are listed in unedited form.	4

Figure 122. IEHLIST Example Directory

Examples that use **disk** in place of actual device numbers must be changed before use. See "DASD and Tape Device Support" on page 3 for valid device number notation.

IEHLIST EXAMPLE 1

In this example, an OS CVOL named SYSCTLG, residing on a disk volume (111111), is listed.

The example follows:

```
//CATLIST  JOB  09#550,BLUE
//         EXEC PGM=IEHLIST
//SYSPRINT DD   SYSOUT=A
//DD2      DD   UNIT=disk,VOLUME=SER=111111,DISP=OLD
//SYSIN    DD   *
     LISTCTLG   VOL=disk=111111
/*
```

The control statements are discussed below:

- DD2 DD defines a mountable device on which the volume containing the source OS CVOL is mounted.
- SYSIN DD defines the control data set, which follows in the input stream.
- LISTCTLG defines the source volume and specifies the list operation.

IEHLIST EXAMPLE 2

In this example, an OS CVOL residing on the system residence volume, two OS CVOLs residing on disk volumes, and a portion of an OS CVOL residing on another volume, are listed.

```
//CATLIST  JOB  09#550,BLUE
//         EXEC PGM=IEHLIST
//SYSPRINT DD   SYSOUT=A
//DD1      DD   UNIT=diskB,VOLUME=SER=111111,DISP=OLD
//DD2      DD   UNIT=(diskA,,DEFER),DISP=OLD,
//               VOLUME=(PRIVATE,,SER=(222222))
//SYSIN    DD   *
     LISTCTLG
     LISTCTLG   VOL=diskA=333333
     LISTCTLG   VOL=diskA=444444
     LISTCTLG   VOL=diskA=555555,NODE=A.B.C
/*
```

The control statements are discussed below:

- DD1 DD defines a system residence device. (The first OS CVOL to be listed resides on the system residence volume.)
- DD2 DD defines a mountable device on which each **diskA** volume is mounted as it is required by the program.
- SYSIN DD defines the control data set, which follows in the input stream.
- The first LISTCTLG statement indicates that the OS CVOL residing on the system residence volume is to be listed.

- The second and third LISTCTLG statements identify two **diskA** disk volumes containing OS CVOLs to be listed.
- The fourth LISTCTLG statement identifies a **diskA** volume containing an OS CVOL that is to be partially listed. All data set entries whose beginning qualifiers are "A.B.C" are listed.

IEHLIST EXAMPLE 3

In this example, a partitioned data set directory existing on the system residence volume is listed. In addition, two partitioned data set directories existing on another disk volume are listed.

```
//LISTPDIR JOB  09#550,BLUE
//         EXEC PGM=IEHLIST
//SYSPRINT DD   SYSOUT=A
//DD1      DD   UNIT=diskB,VOLUME=SER=111111,DISP=OLD
//DD2      DD   UNIT=diskA,VOLUME=SER=222222,DISP=OLD
//SYSIN    DD   *
      LISTPDS   DSNAME=PARSET1
      LISTPDS   DSNAME=(PART1,PART2),VOL=diskA=222222
/*
```

The control statements are discussed below:

- DD1 DD defines the system residence device.
- DD2 DD defines a mountable device on which a disk volume (222222) is to be mounted.
- SYSIN DD defines the control data set, which follows in the input stream.
- The first LISTPDS statement indicates that the partitioned data set directory belonging to data set PARSET1 is to be listed. This data set exists on the system residence volume.
- The second LISTPDS statement indicates that partitioned data set directories belonging to data sets PART1 and PART2 are to be listed. These data sets exist on a disk volume (222222).

IEHLIST EXAMPLE 4

In this example, a non-indexed volume table of contents is listed in the first edited format. The edited listing is supplemented by an unedited listing of selected data set control blocks.

```
//VTOCLIST JOB  09#550,BLUE
//         EXEC PGM=IEHLIST
//SYSPRINT DD   SYSOUT=A
//DD2      DD   UNIT=disk,VOLUME=SER=111111,DISP=OLD
//SYSIN    DD   *
     LISTVTOC   FORMAT,VOL=disk=111111
     LISTVTOC   DUMP,VOL=disk=111111,DSNAME=(SET1,SET2,SET3)
/*
```

The control statements are discussed below:

- DD2 DD defines a mountable device on which the volume containing the specified volume table of contents is to be mounted.
- SYSIN DD defines the control data set, which follows in the input stream.
- The first LISTVTOC statement indicates that the volume table of contents on the specified disk volume is to be listed in edited form.
- The second LISTVTOC statement indicates that the data set control blocks representing data sets SET1, SET2, and SET3 are to be listed in unedited form.

IEHMOVE is a system utility used to move or copy logical collections of operating system data.

IEHMOVE can be used to move or copy:

- A non-VSAM, non-ISAM data set residing on from one to five volumes.
- A group of non-VSAM data sets cataloged in an OS CVOL, ICF or VSAM catalog.
- An entire OS CVOL or portions of an OS CVOL.
- A volume of data sets.
- BDAM data sets with variable-spanned records.

A move operation differs from a copy operation in that a move operation scratches source data if the data set resides on a DASD source volume and the expiration date has occurred, while a copy operation leaves source data intact. In addition, for cataloged data sets, a move operation updates the OS CVOL to refer to the moved version (unless otherwise specified), while a copy operation leaves the OS CVOL unchanged.

The scope of a basic move or copy operation can be enlarged by:

- Including or excluding data sets from a move or copy operation.
- Merging members from two or more partitioned data sets.
- Including or excluding selected members.
- Renaming moved or copied members.
- Replacing selected members.

When moving or copying a data set group or a volume containing password-protected data sets, the user must provide the password each time a data set is opened or scratched.

IEHMOVE always moves or copies any user labels associated with an input data set. IEHMOVE does not take exits to a user's label processing routines.

A move or copy operation results in: (1) a moved or copied data set, (2) no action, or (3) an unloaded[5] version of the source data set. These results depend upon the compatibility of the source and receiving volumes with respect to:

- Size of the volumes
- Allocation of space on the receiving volume
- Data set organization (sequential, partitioned, or BDAM)
- Movability of the source data set

---

[5] If IEHMOVE is unable to successfully move or copy specified data, an attempt is made to reorganize the data and place it on the specified output device. The reorganized data—called an **unloaded data set**—is a sequential data set consisting of 80-byte blocked records that contain the source data and control information for subsequently reconstructing the source data as it originally existed.

VOLUME SIZE COMPATIBILITY

Two volumes are compatible with respect to size if:

1. The source record size does not exceed the receiving track size, or

2. The receiving volume supports the track overflow feature and the output is to be written with track overflow. (Refer to "Job Control Language for the Track Overflow Feature" on page 378 for notes on the track overflow feature.)

When using BDAM data set organization, two volumes are compatible with respect to size if the source track capacity does not exceed the receiving track capacity. BDAM data sets moved or copied to a smaller device type or tape are unloaded. If the user wishes to load an unloaded data set, it must be loaded to the same device type from which it was originally unloaded.

Figure 123 shows the results of move and copy operations when the receiving volume is a DASD volume that is compatible in size with the source volume. The organization of the source data set is shown along with the characteristics of the receiving volume.

Receiving Volume Characteristics	Sequential Data Sets	Partitioned Data Sets	BDAM Data Sets
Space allocated by IEHMOVE (movable data)	Moved or copied	Moved or copied	Moved or copied
Space allocated by IEHMOVE (unmovable data)	Moved or copied	Moved or copied	No action
Space previously allocated, as yet unused	Moved or copied	Moved or copied	No action
Space previously allocated, partially used	No action	Moved or copied (merged)	No action

Figure 123. Move and Copy Operations—DASD Receiving Volume with Size Compatible with Source Volume

Figure 124 shows the results of move and copy operations when the receiving volume is a DASD volume that is not compatible in size with the source volume. The organization of the source data set is shown along with the characteristics of the receiving volume.

Receiving Volume Characteristics	Sequential Data Sets	Partitioned Data Sets	BDAM Data Sets
Space allocated by IEHMOVE	Unloaded	Unloaded	Unloaded
Space previously allocated, as yet unused	Unloaded	Unloaded	No action

Figure 124 (Part 1 of 2). Move and Copy Operations—DASD Receiving Volume with Size Incompatible with Source Volume

Receiving Volume Characteristics	Sequential Data Sets	Partitioned Data Sets	BDAM Data Sets
Space previously allocated, partially used	No action	No action	No action

Figure 124 (Part 2 of 2). Move and Copy Operations—DASD Receiving Volume with Size Incompatible with Source Volume

Figure 125 shows the results of move and copy operations when the receiving volume is not a DASD volume. The organization of the source data set is shown along with the characteristics of the receiving volume.

Receiving Volume Characteristics	Sequential	Partitioned	BDAM
Movable data	Moved or copied	Unloaded	Unloaded
Unmovable data	Unloaded	Unloaded	No action

Figure 125. Move and Copy Operations—Non-DASD Receiving Volume

SPACE ALLOCATION

Space can be allocated for a data set on a receiving volume either by the user (through the use of DD statements in a prior job step) or by IEHMOVE in the IEHMOVE job step. If the source data is unmovable (that is, if it contains location-dependent code), the user should allocate space on the receiving volume using absolute track allocation to ensure that the data set is placed in the same relative location on the receiving volume as it was on the source volume. Unmovable data can be moved or copied if space is allocated by IEHMOVE, but the data may not be in the same location on the receiving volume as it was on the source volume. When data sets are to be moved or copied between unlike DASD devices, a secondary allocation should be made to ensure that ample space is available on the receiving volume.

Space for a new data set should not be allocated by the user when a BDAM data set is to be moved or copied, not unloaded, because IEHMOVE cannot determine if the new data set is empty.

If IEHMOVE performs the space allocation for a new data set, the space requirement information of the old data set (if available) is used. This space requirement information is obtained from the DSCB of the source data set, if it is on a DASD volume, or from the control information in the case of an unloaded data set.

If space requirement information is available, IEHMOVE uses this information to derive an allocation of space for the receiving volume, taking into account the differences in device characteristics, such as track capacity and overhead factors. However, when data sets with variable or undefined record formats are being moved or copied between unlike DASD devices, no assumption can be made about the space that each individual record needs on the receiving device.

In general, when variable or undefined record formats are to be moved or copied, IEHMOVE attempts to allocate sufficient space.

This might cause too much space to be allocated under the following circumstances:

- When moving or copying from a device with a relatively large block overhead to a device with a smaller block overhead, the blocks being small in relation to the block size.
- When moving or copying from a device with a relatively small block overhead to a device with a larger block overhead, the blocks being large in relation to the block size.

BDAM data sets with variable or undefined record formats always have the same amount of space allocated by IEHMOVE. This practice preserves any relative track addressing system that might exist within the data sets.

If a sequential data set, which is not an unloaded data set, on a non-DASD volume is to be moved or copied to a DASD volume, and space attributes are not available through a previous allocation, IEHMOVE makes a default space allocation. The default allocation consists of a primary allocation of 72,500 bytes of DASD storage (data and gaps) and up to 15 secondary allocations of 36,250 bytes each.

Space cannot be previously allocated for a partitioned data set that is to be unloaded unless the SPACE parameter in the DD statement making the allocation implies sequential organization. BDAM data sets should not be previously allocated because IEHMOVE cannot determine whether they are empty or not.

If a move or copy operation is unsuccessful, the source data remains intact.

If a move or copy operation is unsuccessful and space was allocated by IEHMOVE, all data associated with that operation is scratched from the receiving DASD volume. If the receiving volume was tape, it will contain a partial data set.

If a move or copy operation is unsuccessful and space was previously allocated, no data is scratched from the receiving volume. If, for example, IEHMOVE moved 104 members of a 105-member partitioned data set and encountered an input/output error while moving the 105th member:

- The entire partitioned data set is scratched from the receiving volume if space was allocated by IEHMOVE.
- No data is scratched from the receiving volume if space was previously allocated. In this case, after determining the nature of the error, the user need move only the 105th member into the receiving partitioned data set.

If a data set that has only user trailer labels is to be moved from a tape volume to a DASD volume, space must be previously allocated on the DASD volume to ensure that a track is reserved to receive the user labels.

REBLOCKING DATA SETS

Data sets with fixed or variable records can be reblocked to a different block size by previously allocating the desired block size on the receiving volume. No reblocking can be performed when loading or unloading. Also, no reblocking can be performed on data sets with variable spanned or variable blocked spanned records.

When moving or copying data sets with undefined record format and reblocking to a smaller block size (that is, transferring records to a device with a track capacity smaller than the track capacity of the original device), you must make the block size for the receiving volume equal to or larger than the size of the largest record in the data set being moved or copied.

Blocked format data sets that do not contain user data TTRNs or keys can be reblocked or unblocked by including the proper keyword subparameters in the DCB operand of the DD statement used to previously allocate space for the data set. The new blocking factor must be a multiple of the logical record length originally assigned to the data set. For a discussion of user data TTRNs, refer to Data Management Services.

USING IEHMOVE WITH RACF

If the Resource Access Control Facility (RACF) is active, the following considerations apply:

- You must have valid RACF authorization to access any RACF-defined data sets with IEHMOVE. ALTER authorization is required to access the source data set for a MOVE function, as the source data set is scratched. When moving a volume or group of data sets, the user must have adequate access authorization to all of the RACF-protected data sets on the volume or in the group.
- If you have the RACF ADSP attribute and IEHMOVE is to allocate space for the receiving data set, that data set will be automatically defined to RACF. If the data set does not have your userid as the first level qualifier, at least one of the following conditions must be met:
  - You specify MOVE or COPY with RENAME so that the first level qualifier is the correct userid
  - The data set being moved or copied is a group data set and You are connected to the group with CREATE authority
  - You have the OPERATION attribute
- If COPYAUTH is specified and the input data set is RACF-protected (whether or not the user has the ADSP attribute) and the output data set is not preallocated, then the receiving data set of a MOVE or COPY operation is given a copy of the input data set's RACF protection and access list during allocation, governed by the same restrictions described above for defining a data set for a user with the ADSP attribute. The user must have ALTER access authorization to the input data set to either MOVE or COPY using COPYAUTH.

MOVING OR COPYING A DATA SET

IEHMOVE can be used to move or copy sequential, partitioned, and BDAM data sets, as follows:

- A sequential data set can be:
  1. Moved from one DASD volume or non-DASD volume to another (or to the same volume provided that it is a DASD volume), or
  2. Copied from one volume to another (or to the same volume provided that the data set name is changed and the receiving volume is a DASD volume).
- A partitioned data set can be:
  1. Moved from one DASD volume to another (or to the same volume), or
  2. Copied from one DASD volume to another (or to the same volume provided that the data set name is changed).
- A BDAM data set can be moved or copied from one DASD volume to another provided that the receiving device type is the

same device type or larger, and that the record size does not exceed 32K bytes.

SEQUENTIAL DATA SETS

Figure 126 shows basic and optional move and copy operations for sequential data sets.

Operation	Basic Actions	Optional Actions
Move Sequential	Move the data set. For DASD, scratch the source data. For non-VSAM cataloged data sets, update the appropriate catalog to refer to the moved data set.	Prevent automatic cataloging of the moved data set. Rename the moved data set.
Copy Sequential	Copy the data set. The source data set is not scratched. The catalog is not updated to refer to the copied data set.	Delete the catalog or OS CVOL entry for the source data set. Catalog the copied data set on the receiving volume. Rename the copied data set.

Figure 126. Moving and Copying Sequential Data Sets

When moving or copying sequential data sets on DASD, IEHMOVE execution time can be reduced by using multiple BSAM buffers for input and output.

The minimum number of buffers required for enhanced IEHMOVE copy performance is 4: two for input and two for output. The size of an input buffer is computed as: (INPUT BLOCKSIZE + KEY LENGTH) + DECB LENGTH + 4. The size of an output buffer is computed as: (OUTPUT BLOCKSIZE + KEY LENGTH) + DECB LENGTH + 4 + 16.

The maximum number of input buffers used by IEHMOVE is two times the number of buffers which will fit in the input track size. The maximum number of output buffers used by IEHMOVE is two times the number of buffers which will fit in the output track size.

If space for the minimum four buffers is not available, a single buffer is used and message IEH476I is issued.

You can code the JCL REGION parameter in the JOB or EXEC statement to control buffer storage allocation. For details on how to code the REGION parameter. see JCL.

Message IEH477I, describing the number and size of your buffers, will be issued each time multiple BSAM buffers are used. If you do not specify your region size to achieve the maximum number of buffers, the last line of the message will indicate the amount by which the value of the REGION parameter should be increased in order to obtain the maximum number of buffers.

The execution time of an IEHMOVE move or copy operation will vary with the number of buffers available, the size of the data sets, and the block size.

PARTITIONED DATA SETS

Figure 127 shows basic and optional move and copy operations for partitioned data sets.

Operation	Basic Actions	Optional Actions
Move Partitioned	Move the data set. Scratch the source data. For non-VSAM cataloged data sets, update the appropriate catalog to refer to the moved data set.	Prevent automatic cataloging of the moved data set. Rename the moved data set. Reallocate directory space. (Not possible if the space was not allocated by IEHMOVE during this move function.) Perform a merge operation using members from two or more data sets. Move only selected members. Replace members. Unload the data set.
Copy Partitioned	Copy the data set. The source data is not scratched. The catalog is not updated to refer to the copied data set.	Delete the catalog or OS CVOL entry for the source data set. Catalog the copied data set. Rename the copied data set. Reallocate directory space. (Not possible if the space previously allocated is partially used.) Perform a merge operation using members from two or more data sets. Copy only selected members. Replace members. Unload the data set.

Figure 127. Moving and Copying Partitioned Data Sets

IEHMOVE moves or copies partitioned members in the order in which they appear in the partitioned directory. That is, moved or copied members are placed in collating sequence on the receiving volume.

Figure 128 on page 368 shows a copied partitioned data set. The members are copied in the order in which they appear in the partitioned directory. The IEBCOPY utility program (see "IEBCOPY Program" on page 39) can be used to copy data sets whose members are not to be collated.

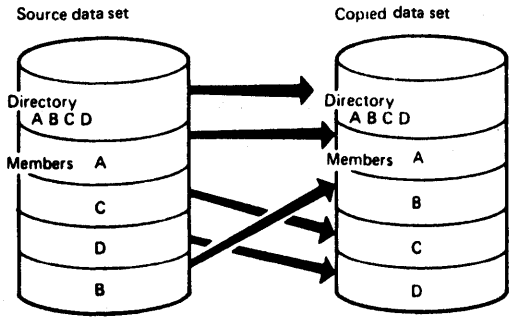


Figure 128. Partitioned Data Set Before and After an IEHMOVE Copy Operation

Members that are merged into an existing data set are placed, in collating sequence, after the last member in the existing data set. If the target data set contains a member with the same name as the data set to be moved, the member will not be moved/copied unless the REPLACE statement is coded.

Figure 129 shows members from one data set merged into an existing data set. Members B and F are copied in collating sequence.

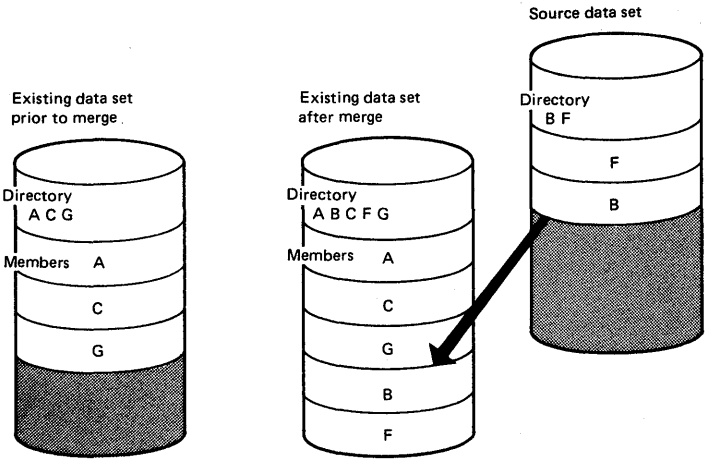


Figure 129. Merging Two Data Sets Using IEHMOVE

Figure 130 on page 369 shows how members from two data sets are merged into an existing data set. Members from additional data sets can be merged in a like manner. Members F, B, D, and E from the source data sets are copied in collating sequence.

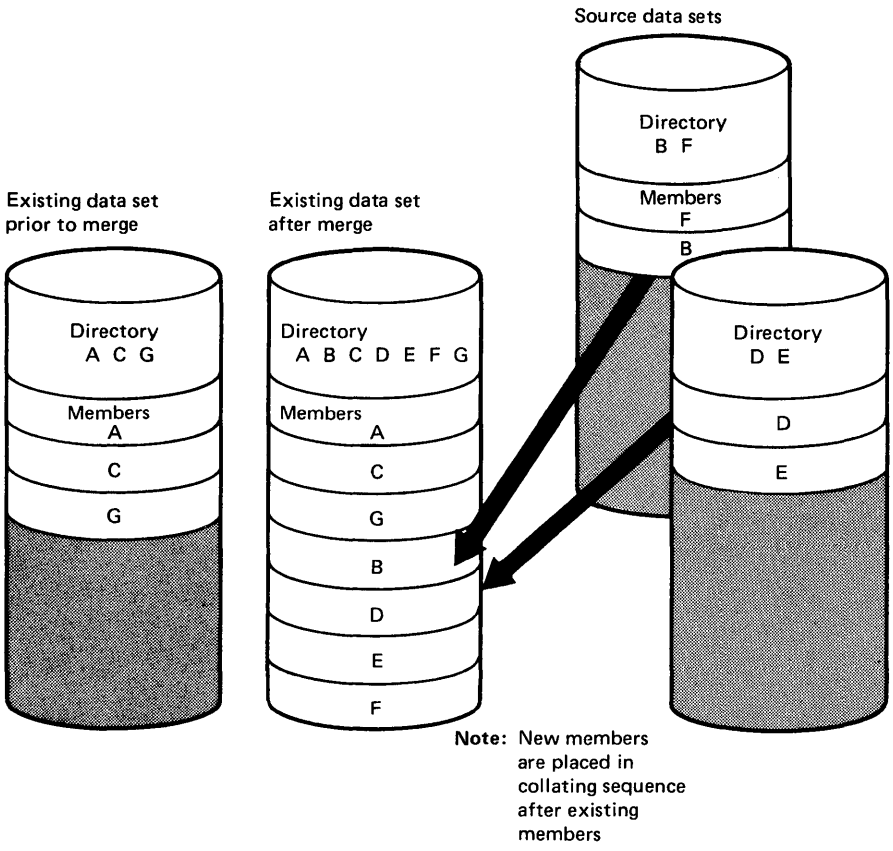


Figure 130. Merging Three Data Sets Using IEHMOVE

BDAM DATA SETS

When moving or copying a BDAM data set from one device to another device of the same type, relative track and relative block integrity are maintained.

When moving or copying a BDAM data set to a larger device, relative track integrity is maintained for data sets with variable or undefined record formats; relative block integrity is maintained for data sets with fixed record formats.

When moving or copying a BDAM data set to a smaller device or a tape, the data set is unloaded. An unloaded data set is loaded only when it is moved or copied to the same device type from which it was unloaded.

MULTIVOLUME DATA SETS

IEHMOVE can be used to move or copy multivolume data sets. To move or copy a multivolume data set, specify the complete volume list in the VOL=SER parameter on the DD statement. A maximum of 5 volumes can be specified. To move or copy a data set that resides on more than one tape volume, specify the volume serial numbers of all the tape volumes and the sequence numbers of the data set on the tape volumes in the utility control statement. (You can specify the sequence number even if the data set to be

moved or copied is the only data set on a volume.) To move or copy a data set to more than one tape volume, specify the volume serial numbers of all the receiving volumes in the utility control statement.

UNLOADED DATA SETS

If IEHMOVE is unable to successfully move or copy specified data, an attempt is made to reorganize the data and place it on the specified output device. The reorganized data—called an **unloaded data set**—is a sequential data set consisting of 80-byte blocked records that contain the source data and control information for subsequently reconstructing the source data as it originally existed.

When an unloaded data set is moved or copied (via IEHMOVE) to a device that will support the data in its true form, the data is automatically reconstructed. For example, if the user attempts to move a partitioned data set to a tape volume, the data is unloaded to that volume. The user can re-create the data set simply by moving the unloaded data set to a DASD volume.

UNMOVABLE DATA SETS

A data set with the unmovable attribute can be moved or copied from one DASD volume to another or to the same volume provided that space has been previously allocated on the receiving volume. Change the name of the data set if move or copy is to be done to the same volume. SVCLIB can be moved or copied to another location on the system residence volume, provided that space has been previously allocated on that volume. The IEHPROGM utility program (see "IEHPROGM Program" on page 404) must be used immediately after such a move operation to rename the moved version SYS1.SVCLIB; If the operation was a 'copy', IEHPROGM must be used to scratch the old version and to rename the copied version.

MOVING OR COPYING A GROUP OF CATALOGED DATA SETS

IEHMOVE can be used to move or copy a group of non-VSAM data sets that are cataloged in ICF or VSAM catalogs and whose names are qualified by one or more identical names. For example, a group of data sets qualified by the name A.B can include data sets named A.B.D and A.B.E, but could not include data sets named A.C.D or A.D.F.

If the user specifies that the data set group is cataloged in an OS CVOL, two additional options are available. First, additional data sets not belonging to the specified data set group can be included in the move or copy operation. Second, data sets belonging to the group can be excluded from the requested operation.

Before copying/moving a DSGROUP that is cataloged in an OS CVOL, the volume containing the OS CVOL must be defined in the ICF or VSAM master catalog. See Catalog Users Guide for details on how this is done.

If a group of data sets is moved or copied to magnetic tape, the data sets must be retrieved one by one by data set name and file-sequence number, or by file-sequence number for unlabeled or nonstandard labeled tapes.

Access method services can be used to determine the structure of ICF or VSAM catalogs. See Access Method Services Reference for more information.

Figure 131 shows basic and optional move and copy operations for a group of non-VSAM cataloged data sets.

Operation	Basic Actions	Optional Actions
Move group of non-VSAM cataloged data sets	Move the data set group (excluding password-protected data sets) to the specified volumes. Scratch the source data sets (BDAM only). Merging is not done.	Prevent updating of the appropriate catalog. Include password-protected data sets in the operation. Unload data sets. If a data set group is cataloged in an OS CVOL, you may INCLUDE or EXCLUDE data sets during the operation.
Copy group of non-VSAM cataloged data sets	Copy the data set group (excluding password-protected data sets). Source data sets are not scratched. Merging is not done.	Include password-protected data sets in the operation. Delete catalog entries for the source data sets. Catalog the copied data sets on the receiving volumes. Unload a data set or sets. If a data set group is cataloged in an OS CVOL, you may INCLUDE or EXCLUDE data sets during the operation.

Figure 131. Moving and Copying a Group of Non-VSAM Cataloged Data Sets

MOVING OR COPYING AN OS CVOL

IEHMOVE can be used to move or copy an OS CVOL or portions of an OS CVOL without copying the data sets represented by the cataloged entries. If the OS CVOL is in an unloaded form, all entries are moved or copied. The SYSCTLG (system catalog) data set need not be defined on the receiving volume before the operation. If, however, SYSCTLG was defined before the operation, the data set organization must not have been specified in the DCB field. Moved or copied entries are merged with any existing entries on the receiving volume. The receiving volume must be a DASD volume unless the OS CVOL is to be unloaded.

Figure 132 shows basic and optional move and copy operations for the OS CVOL.

Operation	Basic Actions	Optional Actions
Move OS CVOL	Move entries from the OS CVOL to the specified DASD volume. Scratch the last index of all entries in the source OS CVOL.	Exclude selected entries from operation. Move an unloaded version of the OS CVOL. Unload the OS CVOL.

Figure 132 (Part 1 of 2). Moving and Copying the OS CVOL

Operation	Basic Actions	Optional Actions
Copy OS CVOL	Copy entries from the OS CVOL to the specified DASD. The source OS CVOL is not scratched.	Exclude selected entries from the operation. Copy an unloaded version of the OS CVOL. Unload the OS CVOL.

Figure 132 (Part 2 of 2). Moving and Copying the OS CVOL

Before copying/moving an OS CVOL, both the volume containing the OS CVOL and the volume to which the OS CVOL is to be moved must be defined in the ICF or VSAM master catalog.

MOVING OR COPYING A VOLUME OF DATA SETS

IEHMOVE can be used to move or copy the data sets of an entire DASD volume to another volume or volumes. A move operation differs from a copy operation in that the move operation scratches source data sets, while the copy operation does not. For both operations, any cataloged entries associated with the source data sets remain unchanged. The IEHPROGM utility program can be used to delete OS CVOL entries for all of the cataloged data sets and recatalog them according to their new location. (See "IEHPROGM Program" on page 404.)

If the source volume contains a SYSCTLG data set, that data set is the last to be moved or copied onto the receiving volume.

If a volume of data sets is moved or copied to tape, sequential data sets are 'moved' while partitioned and BDAM data sets are 'unloaded'. The data sets must be retrieved one by one by data set name and file-sequence number, or by file-sequence number for unlabeled or nonstandard labeled tapes.

When copying a volume of data sets, the user has the option of cataloging all source data sets in a SYSCTLG data set on a receiving volume. However, if a SYSCTLG data set exists on the source volume, error messages indicating that an inconsistent index structure exists are generated when the source SYSCTLG entries are merged into the SYSCTLG data set on the receiving volume.

The move-volume feature does not merge partitioned data sets. If a data set on the volume to be moved has a name identical to a data set name on the receiving volume, the data set is not moved or merged onto the receiving volume.

The copy-volume feature **does** merge partitioned data sets. If a data set on the volume to be copied has a name identical to a data set name on the receiving volume, the data set is copied and merged onto the receiving volume.

Figure 133 shows basic and optional move and copy operations for a volume of data sets.

Operation	Basic Actions	Optional Actions
Move a volume of data sets	Move all data sets not protected by a password to the specified DASD volumes. Scratch the source data sets for DASD volumes. The OS CVOL is not updated.	Include password-protected data sets in the operation. Unload the data sets.
COPY a volume of data sets	Copy all data sets not protected by a password to the specified DASD volume. The source data sets are not scratched.	Include password-protected data sets in the operation. Catalog all copied data sets in the OS CVOL. Unload the data sets.

Figure 133. Moving and Copying a Volume of Data Sets

MOVING OR COPYING BDAM DATA SETS WITH VARIABLE-SPANNED RECORDS

IEHMOVE can be used to move or copy BDAM data sets with variable spanned records from one DASD volume to a compatible DASD volume, provided that the record size does not exceed 32K bytes. (See "Volume Size Compatibility" on page 362 for information on volume compatibility.)

Because a BDAM data set can reside on one to five volumes (all of which must be mounted during any move or copy operation), it is possible for the data set to span volumes. However, single variable-spanned records are contained on one volume.

Relative track integrity is preserved in a move or copy operation for spanned records. Moved or copied BDAM data sets occupy the same relative number of tracks that they occupied on the source device.

If a BDAM data set is unloaded (moved or copied to a smaller device or tape), it must be loaded back to the same device type from which it was originally unloaded.

When moving or copying variable-spanned records to a larger device, record segments are combined and respanned if necessary. Because the remaining track space is available for new records, variable-spanned records are unloaded before being moved or copied back to a smaller device.

If you wish to create a BDAM data set without using data management BDAM macros, all data management specifications must be followed. Special attention must be given to data management specifications for R0 track capacity record content, segment descriptor words, and the BFTEK=R parameter. See Data Management Services for more information on using data management specifications.

When moving or copying a multivolume data set, the secondary allocation for BDAM data sets should be at least two tracks. (See the "WRITE" macro in Data Management Macro Instructions.)

INPUT AND OUTPUT

IEHMOVE uses the following input:

- One or more data sets, which contain the data to be moved, copied, or merged into an output data set.

- A control data set, which contains utility control statements that are used to control the functions of the program.
- A work data set, which is a work area used by IEHMOVE.

IEHMOVE does not support VIO (virtual input/output) data sets.

IEHMOVE produces the following output:

- An output data set, which is the result of the move, copy, or merge operation.
- A message data set, which contains informational messages (for example, the names of moved or copied data sets) and error messages, if applicable.

RETURN CODES

IEHMOVE returns a code in register 15 to indicate the results of program execution. The return codes and their meanings are listed below.

Code	Meaning
00 (00 hex)	Successful completion.
04 (04)	A specified function was not completely successful. Processing continues.
08 (08)	A condition exists from which recovery is possible. Processing continues.
12 (0C)	An unrecoverable error exists. The job step is terminated.
16 (10)	It is impossible to OPEN the SYSIN or SYSPRINT data set. The job step is terminated.

Figure 134. IEHMOVE Return Codes

CONTROL

IEHMOVE is controlled by job control statements and utility control statements. The job control statements are used to execute or invoke the program, define the devices and volumes used and produced by IEHMOVE, and prevent data sets from being deleted inadvertently.

Utility control statements are used to control the functions of the program and to define those data sets or volumes that are to be used.

JOB CONTROL STATEMENTS

Figure 135 on page 375 shows the job control statements for IEHMOVE.

Statement	Use
JOB	Initiates the job.
EXEC	Specifies the program name (PGM=IEHMOVE) or, if the job control statements reside in a procedure library, the procedure name. This statement can include optional PARM information; see "PARM Information on the EXEC Statement" on page 375.
SYSPRINT DD	Defines a sequential message data set. The data set can be written onto a system output device, a magnetic tape volume, or a direct access volume.
SYSUT1 DD	Defines a volume on which three work data sets required by IEHMOVE are allocated.
anyname1 DD	Defines a permanently mounted DASD volume. (The system residence volume is considered to be a permanently mounted volume.) This statement is required.
anyname2 DD	Defines a mountable device type. At least one anyname2 DD statement is required. Multiple statements must have unique names.
tape DD	Defines a mountable tape device.
SYSIN DD	Defines the control data set. The data set, which contains utility control statements, usually follows the job control statements in the input stream; however, it can be defined either as a sequential data set or as a member of a procedure library.

Figure 135. IEHMOVE Job Control Statements

PARM Information on the EXEC Statement

The EXEC statement for IEHMOVE can contain PARM information that is used by the program to allocate additional work space and/or control line density on output listings. The EXEC statement can be coded, as follows:

//	**EXEC**	**PGM=IEHMOVE[,PARM='POWER=n' [,'LINECNT=xx']]**

The POWER=n parameter is used to request that the normal amount of space allocated for work areas be increased n times (1 to 999). The POWER parameter is used when 750 or more members are being moved or copied. The progression for the value of n is:

- POWER=2 when 750 to 1,500 members are to be moved or copied.
- POWER=3 when 1,501 to 2,250 members are to be moved or copied.
- POWER=4 when 2,251 to 3,000 members are to be moved or copied.

If POWER=2, the work space requirement on the SYSUT1 volume is two times the basic requirement; if POWER=3, work space requirement is three times the basic requirement, etc. For example, if POWER=2, three areas of 26, 26, and 52 contiguous tracks on a 3380 must be available.

When moving or copying an OS CVOL, the value of the POWER parameter can be calculated, as follows:

$$n=(10D + V + 20G)/4000$$

where D is the total number of data sets, aliases, and generation data set entries (which is the number of data set names printed by the IEHLIST utility program when the LISTCTLG statement is specified); V is the total number of volumes used by these data sets (which is the number of lines printed by the IEHLIST utility program when the LISTCTLG statement is specified); and G is the number of generation data sets. Approximate values can be used:

- POWER=2 when 350 to 700 data sets are cataloged.
- POWER=3 when 701 to 1,050 data sets are cataloged.
- POWER=4 when 1,051 to 1,400 data sets are cataloged.

The LINECNT=xx parameter specifies the number of lines per page in the listing of the SYSPRINT data set; xx is a two-digit number in the range 04 through 99.

See Supervisor Services and Macro Instructions for more information on PARM values.

SYSPRINT DD Statement

The block size for the SYSPRINT data set must be a multiple of 121. Any blocking factor can be specified.

SYSUT1 DD Statement

The SYSUT1 DD statement must be coded:

```
//SYSUT1 DD UNIT=xxxx,VOLUME=SER=xxxxxx,DISP=OLD
```

The UNIT and VOLUME parameters define the device type and volume serial number. The DISP=OLD specification prevents the inadvertent deletion of a data set.

At least three utility work areas of 13, 13, and 26 contiguous tracks, respectively, must be available for work space on the volume defined by the SYSUT1 DD statement. (This figure is based on a 3380 being the work volume. If a direct access device other than a 3380 is used, an equivalent amount of space must be available.)

anyname1 DD Statement

One anyname1 DD statement must be included for each permanently mounted volume referred to in the job step.

The anyname1 DD statement should be coded:

```
//anyname1 DD UNIT=xxxx,VOLUME=SER=xxxxxx,DISP=OLD
```

In the anyname1 DD statement, the UNIT and VOLUME parameters define the device type and volume serial number. The DISP=OLD specification prevents the inadvertent deletion of a data set.

When unloading a data set from one DASD volume to another, the data set name (DSN=) must be coded on the DD statement for the data set to be unloaded. An unloaded data set on a DASD volume can only be loaded to the same device type from which it was unloaded.

(The anyname1 DD statement is arbitrarily assigned the ddname DD1 in the IEHMOVE examples.)

anyname2 DD Statement

One anyname2 DD statement must be included for each mountable device to be used in the job step. Multiple anyname2 DD statements must have unique names.

When IEHMOVE is dynamically invoked in a job step containing another program, the DD statements defining mountable devices for IEHMOVE must be included in the job stream prior to DD statements defining data sets required by the other program.

The anyname2 DD statement should be coded:

```
//anyname2 DD UNIT=xxxx,VOLUME=SER=xxxxxx,DISP=OLD
```

The UNIT and VOLUME parameters define the device type and volume serial number. The DISP=OLD specification prevents the inadvertent deletion of a data set.

When unloading a data set from one DASD volume to another, the data set name (DSN=) must be coded on the DD statement for the data set to be unloaded. An unloaded data set on a DASD volume can only be loaded to the same device type from which it was unloaded.

(The anyname2 DD statement is arbitrarily assigned the ddname DD2 in the IEHMOVE examples.)

When the number of volumes to be processed is greater than the number of devices defined by DD statements, there must be an indication (in the applicable DD statements) that multiple volumes are to be processed. This indication can be in the form of deferred mounting, as follows:

```
//anyname2 DD  UNIT=(xxxx,,DEFER),VOLUME=(PRIVATE,...),
//             DISP=(...,KEEP)
```

See Appendix B, "DD Statements for Defining Mountable Devices" on page 443 for information on defining mountable devices. (DD statements defining additional mountable device types are assigned names DD3, DD4, etc., in the IEHMOVE examples.) Unit affinity cannot be used on DD statements defining mountable devices.

A merge operation requires that one DD statement defining a mountable device be present for each source volume containing data to be included in the merge operation.

tape DD Statement

The tape DD statement can be coded:

```
//tape DD  DSNAME=xxxxxxxx,UNIT=xxxx,VOLUME=SER=xxxxxx,
//         DISP=(...,KEEP),LABEL=(...,...),DCB=(TRTCH=C,DEN=x)
```

When unloading a data set from one DASD volume to another, the data set name (DSN=) must be coded on the DD statement for the data set to be unloaded. An unloaded data set on a DASD volume can only be loaded to the same device type from which it was unloaded.

A utility control statement parameter refers to the tape DD statement for label and mode information.

The date on which a data set is moved or copied to a magnetic tape volume is automatically recorded in the HDR1 record of a standard tape label if a TODD parameter is specified in a utility control statement. An expiration date can be specified by including the EXPDT or RETPD subparameters of the LABEL keyword in the DD statement referred to by a TODD parameter.

A sequence number, for a data set on a tape volume, or a specific device address (for example, unit address 190), must be

specified on a utility control statement instead of a reference to a DD statement. To move or copy a data set from or to a tape volume containing more than one data set, specify the sequence number of the data set in the utility control statement. To move or copy a data set from or to a specific device, specify the unit address (rather than a group name or device type) in the utility control statement. To copy to a unit record or unlabeled tape volume, specify any standard name or number in the utility control statement.

The tape DD statement can be used to communicate DCB attributes of data sets residing on tape volumes that do not have standard labels to IEHMOVE. If no DCB attributes are specified, an undefined record format and a block size of 2560 are assumed. However, in order to recognize unloaded data sets on an unlabeled tape volume, the DCB attributes must be specified as follows:

```
DCB=(RECFM=FB,LRECL=80,BLKSIZE=800).
```

IEHMOVE automatically calculates and allocates the amount of space needed for the work areas. No SPACE parameter, therefore, should be coded in the SYSUT1 DD statement. If, in the PARM field of the EXEC statement, POWER=3 is specified, the work space requirement is three times the basic requirements, etc.

Prior space allocations can be made by specifying a dummy execution of the IEHPROGM utility program before the execution of IEHMOVE.

SYSIN DD Statement

The block size for the SYSIN data set must be a multiple of 80. Any blocking factor can be specified.

Job Control Language for the Track Overflow Feature

A data set containing track overflow records can be moved or copied if the source volume and the receiving volume are mounted on DASD that support the track overflow feature. (For BDAM data sets, the source and receiving devices must be the same device type.)

A data set that was written without track overflow can be moved or copied with or without track overflow or vice versa if the following conditions are met:

- Space was allocated for the data set prior to the request for a move or copy operation.
- The DD statement used for that allocation included the subparameter to specify the changed track overflow value and all other desired values. (The RECFM specifications assigned when the data set was originally created are overridden by the RECFM subparameter in this DD statement.)

If space has not been allocated, or if RECFM was not specified when space was allocated, the data set is moved or copied in accordance with RECFM specifications that were made when the data set was originally created. This track overflow attribute is not retained for a sequential data set that is moved or copied to a device other than a DASD.

UTILITY CONTROL STATEMENTS

IEHMOVE is controlled by the following utility control statements.

Statement	Use
MOVE DSNAME	Moves a data set.
COPY DSNAME	Copies a data set.
MOVE DSGROUP	Moves a group of non-VSAM cataloged data sets.
COPY DSGROUP	Copies a group of non-VSAM cataloged data sets.
MOVE PDS	Moves a partitioned data set.
COPY PDS	Copies a partitioned data set.
MOVE VOLUME	Moves a volume of data sets.
COPY VOLUME	Copies a volume of data sets.
MOVE CATALOG	Moves OS CVOL entries.
COPY CATALOG	Copies OS CVOL entries.

Figure 136. IEHMOVE Utility Control Statements

In addition, there are four subordinate control statements that can be used to modify the effect of a MOVE DSGROUP, COPY DSGROUP, MOVE PDS, COPY PDS, MOVE CATALOG, or COPY CATALOG operation. The subordinate control statements are:

- **INCLUDE** statement, which is used to enlarge the scope of a MOVE DSGROUP (with CVOL), COPY DSGROUP (with CVOL), MOVE PDS, or COPY PDS statement by including a member or data set not explicitly included by the statement it modifies.
- **EXCLUDE** statement, which is used with a MOVE DSGROUP (with CVOL), COPY DSGROUP (with CVOL), MOVE PDS, COPY PDS, MOVE CATALOG, or COPY CATALOG statement to exclude data set(s), a member or OS CVOL entry(ies) from a move or copy operation.
- **REPLACE** statement, which is used with a MOVE PDS or COPY PDS statement to exclude a member from a move or copy operation and to replace it with a member from another partitioned data set.
- **SELECT** statement, which is used with MOVE PDS or COPY PDS statements to select members to be moved or copied and, optionally, to rename the specified members.

Continuation requirements for utility control statements are described in "Continuing Utility Control Statements" on page 5.

MOVE DSNAME Statement

The MOVE DSNAME statement is used to move a data set. The source data set is scratched.

If the data set is cataloged (in an OS CVOL, ICF or VSAM catalog), the catalog is automatically updated unless UNCATLG/FROM is specified.

The format of the MOVE DSNAME statement is:

[label]	MOVE	DSNAME=name ,TO=device=list [,{FROM=device=list|CVOL=device=serial}] [,UNCATLG] [,RENAME=name] [,FROMDD=ddname] [,TODD=ddname] [,UNLOAD] [,COPYAUTH]

COPY DSNAME Statement

The COPY DSNAME statement is used to copy a data set.

The source data set, if cataloged, remains cataloged unless UNCATLG or CATLG is specified without the RENAME and FROM parameters.

The format of the COPY DSNAME statement is:

[label]	COPY	DSNAME=name ,TO=device=list [,{FROM=device=list|CVOL=device=serial}] [,UNCATLG] [,CATLG] [,RENAME=name] [,FROMDD=ddname] [,TODD=ddname] [,UNLOAD] [,COPYAUTH]

MOVE DSGROUP Statement

The MOVE DSGROUP statement is used to move groups of data sets whose names are partially qualified by one or more identical names. The data sets may be cataloged on several catalogs (OS CVOL, ICF or VSAM). Source data sets are scratched. Data set groups to be moved must reside on DASD volumes. Only data sets that could be moved by MOVE DSNAME or MOVE PDS can be moved by MOVE DSGROUP. Alias entries in ICF or VSAM catalogs for the data sets are lost and can be replaced with access method services. See Access Method Services Reference for more information.

INCLUDE and EXCLUDE statements, discussed later in this chapter, can be used to add to or delete data sets from the group, if CVOL is specified.

MOVE DSGROUP operations cause the catalog to be updated automatically unless UNCATLG is specified.

The format of the MOVE DSGROUP statement is:

[label]	MOVE	DSGROUP[=name] ,TO=device=list [,CVOL=device=serial] [,PASSWORD] [,UNCATLG] [,TODD=ddname] [,UNLOAD] [,COPYAUTH]

COPY DSGROUP Statement

The COPY DSGROUP statement is used to copy groups of data sets whose names are partially qualified by one or more identical names. The data sets may be cataloged on several catalogs (OS CVOL, ICF or VSAM). Only data sets that can be copied with COPY DSNAME or COPY PDS can be copied with COPY DSGROUP. Data set groups to be copied must reside on DASD volumes.

INCLUDE and EXCLUDE statements, discussed later in this chapter, can be used to add to or delete data sets from the group, if CVOL is specified.

The source data sets remain cataloged unless UNCATLG or CATLG is specified without the RENAME and FROM parameters.

The format of the COPY DSGROUP statement is:

[label]	COPY	DSGROUP[=name] ,TO=device=list [,CVOL=device=serial] [,PASSWORD] [,UNCATLG] [,CATLG] [,TODD=ddname] [,UNLOAD] [,COPYAUTH]

MOVE PDS Statement

The MOVE PDS statement is used to move partitioned data sets. When used in conjunction with INCLUDE, EXCLUDE, REPLACE, or SELECT statements, the MOVE PDS statement can be used to merge selected members of several partitioned data sets or to delete members.

If IEHMOVE is used to allocate space for an output partitioned data set, the MOVE PDS statement can be used to expand a partitioned directory.

If the receiving volume contains a partitioned data set with the same name, the two data sets are merged. The source data set is scratched.

MOVE PDS causes the appropriate catalog to be updated automatically unless UNCATLG/FROM is specified.

The format of the MOVE PDS statement is:

[label]	MOVE	PDS=name ,TO=device=serial|list [,{FROM=device=serial|CVOL=device=serial}] [,EXPAND=nn] [,UNCATLG] [,RENAME=name] [,FROMDD=ddname] [,TODD=ddname] [,UNLOAD] [,COPYAUTH]

COPY PDS Statement

The COPY PDS statement is used to copy partitioned data sets. When used in conjunction with INCLUDE, EXCLUDE, REPLACE, or SELECT statements, the COPY PDS statement can be used to merge selected members of several partitioned data sets or to delete members.

If IEHMOVE is used to allocate space for an output partitioned data set, the COPY PDS statement can be used to expand a partitioned directory.

If the receiving volume already contains a partitioned data set with the same name, the two are merged.

The source partitioned data set remains cataloged unless UNCATLG or CATLG is specified without the RENAME and FROM parameters.

The format of the COPY PDS statement is:

[label]	COPY	PDS=name ,TO=device=serial|list [,{FROM=device=serial|CVOL=device=serial}] [,EXPAND=nn] [,UNCATLG] [,CATLG] [,RENAME=name] [,FROMDD=ddname] [,TODD=ddname] [,UNLOAD] [,COPYAUTH]

MOVE CATALOG Statement

The MOVE CATALOG statement is used to move the entries of an OS CVOL (SYSCTLG data set) without moving the data sets associated with those entries. Certain entries can be excluded from the operation by means of the EXCLUDE statement. If the receiving volume already contains an OS CVOL, the source OS CVOL entries are merged with it.

The format of the MOVE CATALOG statement is:

[label]	MOVE	CATALOG[=name] TO=device=serial|list [,{FROM=device=serial|CVOL=device=serial}] [,FROMDD=ddname] [,TODD=ddname] [,UNLOAD] [,COPYAUTH]

COPY CATALOG Statement

The COPY CATALOG statement is used to copy the entries of an OS CVOL (SYSCTLG data set) without copying the data sets associated with those entries. Certain entries can be excluded from a copy operation with the EXCLUDE statement. If the receiving volume already contains an OS CVOL, the source OS CVOL is merged with it.

The format of the COPY CATALOG statement is:

[label]	COPY	CATALOG[=name] ,TO=device=serial|list [,{FROM=device=serial|CVOL=device=serial}] [,FROMDD=ddname] [,TODD=ddname] [,UNLOAD] [,COPYAUTH]

MOVE VOLUME Statement

The MOVE VOLUME statement is used to move all the data sets residing on a specified volume. Any catalog entries associated with the data sets remain unchanged. Data sets to be moved must reside on DASD volumes.

The format of the MOVE VOLUME statement is:

[label]	MOVE	VOLUME=device=serial ,TO=device=list [,PASSWORD] [,TODD=ddname] [,UNLOAD] [,COPYAUTH]

COPY VOLUME Statement

The COPY VOLUME statement is used to copy all the data sets residing on a specified volume. Any catalog entries associated with the data sets remain unchanged. Data sets to be copied must reside on DASD volumes.

If CATLG and CVOL are specified, error messages indicating that an inconsistent index structure exists are issued when the source SYSCTLG data set entries are merged into the OS CVOL on the receiving volume. (Because the SYSCTLG data set is the last to be copied, only those entries representing cataloged data sets not residing on the source volume are copied into a receiving volume's SYSCTLG data set; entries representing all data sets residing on the source volume have already been made in the receiving SYSCTLG data set.)

The format of the COPY VOLUME statement is:

[label]	COPY	VOLUME=device=serial ,TO=device=list [,PASSWORD] [,CATLG] [,TODD=ddname] [,UNLOAD] [,COPYAUTH]

INCLUDE Statement

The INCLUDE statement is used to enlarge the scope of MOVE DSGROUP, COPY DSGROUP, MOVE PDS, or COPY PDS statements by including a member or a data set not explicitly defined in those statements. The INCLUDE statement follows the MOVE or COPY statement whose function it modifies. The record characteristics of the included partitioned data sets must be compatible with those of the other partitioned data sets being moved or copied. Any number of INCLUDE statements can modify a MOVE or COPY statement. For a partitioned data set, the INCLUDE statement is invalid when data is unloaded or when unloaded data is moved or copied. For DSGROUP operations, INCLUDE is invalid unless CVOL has been specified on the MOVE/COPY DSGROUP control statement.

The format of the INCLUDE statement is:

[label]	INCLUDE	DSNAME=name [,MEMBER=membername] [,{FROM=device=list|CVOL=device=serial}]

EXCLUDE Statement

The EXCLUDE statement is used to restrict the scope of MOVE DSGROUP, COPY DSGROUP, MOVE PDS, COPY PDS, MOVE CATALOG, or COPY CATALOG statements by excluding a specific portion of data defined in those statements.

Partitioned data set members excluded from a MOVE PDS operation cannot be recovered (the source data set is scratched). Any number of EXCLUDE statements can modify a MOVE PDS or COPY PDS statement.

Source data sets or OS CVOL entries excluded from a MOVE DSGROUP or MOVE CATALOG operation remain available. Only one EXCLUDE statement can modify a MOVE DSGROUP, COPY DSGROUP, MOVE CATALOG, or COPY CATALOG statement. The EXCLUDE statement is invalid when data is unloaded or when unloaded data is moved or copied. The EXCLUDE statement is invalid for a DSGROUP operation unless CVOL is specified on the MOVE/COPY DSGROUP control statement.

The format of the EXCLUDE statement is:

[label]	EXCLUDE	{DSGROUP=name|MEMBER=membername}

SELECT Statement

The SELECT statement is used with the MOVE PDS or COPY PDS statement to select members to be moved or copied, and to optionally rename these members. The SELECT statement cannot be used with either the EXCLUDE or REPLACE statement to modify the same MOVE PDS or COPY PDS statement. The SELECT statement is invalid when data is unloaded or when unloaded data is moved or copied. Members not selected in a MOVE PDS operation cannot be recovered since the source data set is scratched.

The format of the SELECT statement is:

[label]	SELECT	{MEMBER=(name[,name]...)| MEMBER=((name,newname)[,(name,newname)]...)

REPLACE Statement

The REPLACE statement is used with a MOVE PDS or COPY PDS statement to exclude a member from the operation and replace it with a member from another partitioned data set. The new member must have the same name as the old member and must possess compatible record characteristics. Any number of REPLACE statements can modify a MOVE PDS or COPY PDS statement. The REPLACE statement is invalid when data is unloaded or when unloaded data is moved or copied.

The format of the REPLACE statement is:

[label]	**REPLACE**	**DSNAME=**name **,MEMBER=**name **[,{FROM=**device**=**serial**|CVOL=**device**=**serial**}]**

Parameters	Applicable Control Statements	Description of Parameters
CATALOG	MOVE CATALOG COPY CATALOG	**CATALOG[=**name**]** specifies the OS CVOL entries to be moved or copied. If name is not coded, all entries in the OS CVOL are moved or copied. If name is coded, all OS CVOL entries whose names are qualified by this name are moved or copied. If the name is a fully qualified data set name, (for example, AAA.BBB.CC), only the OS CVOL entry that corresponds to that data set is moved or copied.
CATLG	COPY DSNAME COPY DSGROUP COPY PDS COPY VOLUME	**CATLG** specifies that the copied data set(s) is (are) cataloged as described below. 1. If the CVOL parameter is omitted, the cataloging is done in the ICF or VSAM master/JOBCAT/STEPCAT catalog. 2. If the RENAME and FROM parameters are omitted, the source data set(s) entry is deleted from the appropriate catalog to permit the copied data set(s) to be recataloged. 3. If the CVOL parameter is specified, the cataloging is done in the OS CVOL on the receiving DASD volume. If an OS CVOL does not exist on the receiving DASD volume, one is created.
COPYAUTH	MOVE DSNAME COPY DSNAME MOVE DSGROUP COPY DSGROUP MOVE PDS COPY PDS MOVE CATALOG COPY CATALOG MOVE VOLUME COPY VOLUME	**COPYAUTH** specifies that the receiving data set is to be given the same access list as the input data set, if the input data set is RACF protected and the output data set is not preallocated.

Parameters	Applicable Control Statements	Description of Parameters
CVOL	MOVE DSNAME COPY DSNAME MOVE PDS COPY PDS INCLUDE REPLACE	**CVOL=**device=serial specifies the device type and serial number of the OS CVOL on which the search for the data set is to begin. If the CVOL or FROM parameter is omitted, the data set is assumed to be cataloged in the ICF or VSAM master/JOBCAT/STEPCAT catalog. FROM and CVOL should never appear in the same utility control statement.
	MOVE DSGROUP COPY DSGROUP	**CVOL=**device=serial specifies the device type and serial number of the OS CVOL on which the search for the data set(s) is to begin. If the CVOL parameter is omitted, the data set(s) is assumed to be cataloged in the ICF or VSAM master/JOBCAT/STEPCAT catalog.
	MOVE CATALOG COPY CATALOG	**CVOL=**device=serial specifies the device type and serial number of the volume from which the SYSCTLG data set is to be moved or copied. If the CVOL or FROM parameter is omitted, the SYSCTLG data set to be moved or copied is assumed to reside on the system residence volume. FROM and CVOL should never appear in the same utility control statement.
DSGROUP	MOVE DSGROUP COPY DSGROUP	**DSGROUP=**name specifies the cataloged data set(s) to be moved or copied. If name is a fully qualified data set name, only that data set is not moved or copied. If name is one or more qualifiers, but not fully qualified, all data sets whose names are qualified by name are moved or copied. If name is omitted, all data sets whose names are found in the searched catalog are moved or copied.
	EXCLUDE	**DSGROUP=**name Specifies the cataloged data set(s) or the catalog entry(ies) to be excluded in a MOVE/COPY DSGROUP or CATALOG operation. If used in conjunction with MOVE/COPY DSGROUP, all cataloged data sets whose names are qualified by name are excluded from the operation. If used in conjunction with MOVE/COPY CATALOG, all catalog entries whose names are qualified by name are excluded from the operation. The CVOL parameter must be specified if a MOVE/COPY DSGROUP operation is being performed.

Parameters	Applicable Control Statements	Description of Parameters
DSNAME	MOVE DSNAME COPY DSNAME	**DSNAME=**name specifies the fully qualified name of the data set to be moved or copied.
	INCLUDE	**DSNAME=**name specifies the fully qualified name of a data set. If used in conjunction with MOVE/COPY DSGROUP, the named data set is included in the group. If used in conjunction with MOVE/COPY PDS, either the named partitioned data set or a member of it (if the MEMBER parameter is specified) is included in the operation.
	REPLACE	**DSNAME=**name specifies the fully qualified name of the partitioned data set that contains the replacement member.
EXPAND	MOVE PDS COPY PDS	**EXPAND=**nn specifies the decimal number (up to 99) of 256-byte records to be added to the directory of the specified partitioned data set. For COPY, EXPAND cannot be specified if space is previously allocated. For MOVE, EXPAND will be ignored if space is previously allocated.

Parameters	Applicable Control Statements	Description of Parameters
FROM	MOVE DSNAME COPY DSNAME MOVE PDS COPY PDS INCLUDE REPLACE MOVE CATALOG COPY CATALOG	**FROM=**device=list|serial specifies the unit address or device type and serial number(s) of the volume(s) on which the data set resides if it is not cataloged. If the data set is cataloged, FROM should not be specified. When FROM is to refer to a specific device, code the unit address in the device parameter, in place of device type. The serial subparameter applies to PDS and CATALOG operations. The list subparameter applies to DSNAME operations, but may also be used when referring to an unloaded PDS residing on more than one DASD or tape volume, and when referring to an unloaded OS CVOL residing on more than one tape volume. When FROM is used in conjunction with a MOVE, DSNAME/PDS operation, the catalog will not be updated. When FROM is used in conjunction with a MOVE/COPY CATALOG operation, it specifies where an unloaded version of the OS CVOL resides. When FROM refers to a tape device and the data set to be retrieved is not the first on the volume, the serial subparameter must be enclosed in parentheses and the volume serial number must be followed by the data set sequence number and separated from it by a comma, as follows: FROM=device=(serial,seqnumber) If FROM or CVOL parameter is omitted from a MOVE/COPY DSNAME/PDS, INCLUDE or REPLACE operation, the data set is assumed to be cataloged in the ICF or VSAM master/JOBCAT/STEPCAT catalog. If the FROM or CVOL parameter is omitted from a MOVE/COPY CATALOG operation, the SYSCTLG data set to be moved or copied is assumed to reside on the system residence volume. FROM and CVOL should never be specified on the same utility control statement.
FROMDD	MOVE DSNAME COPY DSNAME MOVE PDS COPY PDS MOVE CATALOG COPY CATALOG	**FROMDD=**ddname specifies the name of the DD statement from which DCB and LABEL information (except data set sequence number), for input data sets on tape volumes, can be obtained. When FROMDD is used in conjunction with a MOVE/COPY PDS/CATALOG operation, the tape data set must be an unloaded version of a partitioned data set or an unloaded version of an OS CVOL. The FROMDD operand can be omitted, provided the data set has standard labels and resides on a 9-track tape volume.

Parameters	Applicable Control Statements	Description of Parameters
MEMBER	INCLUDE REPLACE	**MEMBER=**membername specifies the name of one member in the partitioned data set named in the DSNAME parameter on the INCLUDE/REPLACE statement. When coded on an INCLUDE statement, the named member is merged with the partitioned data set being moved or copied. When coded on a REPLACE statement, the member replaces an equally named member in the partitioned data set being moved or copied. Regardless of the operation, neither the partitioned data set containing the named member nor the member is scratched.
	EXCLUDE	**MEMBER=**membername specifies the name of a member to be excluded from a MOVE/COPY PDS operation
	SELECT	**MEMBER=**{name|(name[,name]...)|((name,newname)[,(name,newname)]...)} specifies the names of the members to be moved or copied by a MOVE/COPY PDS operation, and optionally new names to be assigned to the members.
PASSWORD	MOVE DSGROUP COPY DSGROUP MOVE VOLUME COPY VOLUME	**PASSWORD** specifies that password protected data sets are included in the operation. This is not VSAM password protection, but the OS password scheme. **Default:** Only data sets that are not protected are copied or moved.
PDS	MOVE PDS COPY PDS	**PDS=**name specifies the fully qualified name (that is, the name with all its qualifiers, if any) of the partitioned data set to be moved or copied.
RENAME	MOVE DSNAME COPY DSNAME MOVE PDS COPY PDS	**RENAME=**name specifies that the data set is to be renamed, and indicates the new name.
TO	MOVE DSNAME COPY DSNAME MOVE DSGROUP COPY DSGROUP MOVE VOLUME COPY VOLUME	**TO=**device=list specifies the device type and volume serial number of the volume or volumes to which the specified group of data sets is to be moved or copied.
	MOVE PDS COPY PDS MOVE CATALOG COPY CATALOG	**TO=**device=serial|list specifies the device type and volume serial number of the volume to which the partitioned data set or OS CVOL entry is to be moved or copied. The list parameter may be used when unloading a partitioned data set that must span tape volumes.

Parameters	Applicable Control Statements	Description of Parameters
TODD	MOVE DSNAME COPY DSNAME MOVE DSGROUP COPY DSGROUP MOVE PDS COPY PDS MOVE VOLUME COPY VOLUME MOVE CATALOG COPY CATALOG	**TODD=**ddname specifies the name of a DD statement from which DCB (except RECFM, BLKSIZE and LRECL) and LABEL (except data set sequence number) information for output data sets on tape volumes can be obtained. When TODD is used in conjunction with a MOVE/COPY DSNAME/DSGROUP/VOLUME operation, it describes the mode and label information to be used when creating output data sets on tape volumes. RECFM, BLKSIZE, and LRECL information, if coded, is ignored. When UNLOAD is specified, or when TODD is used in conjunction with a MOVE/COPY PDS/CATALOG operation, it describes the mode and label information to be used when creating unloaded versions of data sets on tape volumes. RECFM, BLKSIZE, and LRECL information, if coded, must specify (RECFM=FB, BLKSIZE=800, LRECL=80). TODD must be specified in the control statement when an expiration data (EXPDT) or retention period (RETPD) is to be created or changed. The TODD parameter can be omitted for 9-track tapes with standard labels and default density for the unit type specified.
UNCATLG	MOVE DSNAME COPY DSNAME MOVE DSGROUP COPY DSGROUP MOVE PDS COPY PDS	**UNCATLG** specifies that the catalog entry pertaining to the source partitioned data set is to be removed. This parameter should be used only if the source data set is cataloged. If the volume is identified by FROM, UNCATLG is ignored. Alias entries in ICF or VSAM catalogs for the source data sets are lost and can be replaced with access method services if the data sets are later cataloged. See Access Method Services Reference for more information. For a MOVE operation, UNCATLG inhibits cataloging of the output data set.
UNLOAD	MOVE DSNAME COPY DSNAME MOVE DSGROUP COPY DSGROUP MOVE PDS COPY PDS MOVE VOLUME COPY VOLUME MOVE CATALOG COPY CATALOG	**UNLOAD** specifies that the data set is to be unloaded to the receiving volume(s).
VOLUME	MOVE VOLUME COPY VOLUME	**VOLUME=**device=serial specifies the device type and volume serial number of the source volume.

IEHMOVE EXAMPLES

The following examples illustrate some of the uses of IEHMOVE. Figure 137 can be used as a quick reference guide to IEHMOVE examples. The numbers in the "Example" column refer to the examples that follow.

Operation	Data Set Organization	Device	Comments	Example
MOVE	Sequential	Disk	Source volume is demounted after job completion. Two mountable disks.	1
COPY	Sequential	Disk	Three cataloged sequential data sets are copied. The disks are mountable.	2
MOVE	Partitioned	Disk	A partitioned data set is moved; a member from another PDS is merged with it.	3
MOVE	Volume	Disk	A volume of data sets is moved to a disk volume.	4
MOVE	Partitioned	Disk	A data set is moved to a volume on which space was previously allocated.	5
MOVE	Partitioned	Disk	Three data sets are moved and unloaded to a volume on which space was previously allocated.	6
MOVE	Sequential	Disk and Tape	A sequential data set is unloaded to an unlabeled 9-track tape volume.	7
MOVE	Sequential	Disk and Tape	Unloaded data sets are loaded from a single volume.	8
COPY	Sequential	Disk and Tape	Data sets are copied from separate source volumes.	9
COPY	Partitioned	Tape and Disk	Unloaded data sets are loaded from unlabeled tape to a specific device.	10
MOVE	Data Set Group	Disk	Data set group is moved.	11

Figure 137 (Part 1 of 2). IEHMOVE Example Directory

Operation	Data Set Organization	Device	Comments	Example
MOVE	OS CVOL	Disk	SYSCTLG data set (OS CVOL) is moved from one volume to another. Source OS CVOL is scratched.	12
MOVE	OS CVOL	Disk	Selected OS CVOL entries are moved from one OS CVOL to another.	13

Figure 137 (Part 2 of 2). IEHMOVE Example Directory

Examples that use **disk** or **tape** in place of actual device numbers must be changed before use. See "DASD and Tape Device Support" on page 3 for valid device number notation.

IEHMOVE EXAMPLE 1

In this example, three sequential data sets (SEQSET1, SEQSET2, and SEQSET3) are moved from a disk volume to three separate disk volumes. Each of the three receiving volumes is mounted when it is required by IEHMOVE. The source data sets are not cataloged. Space is allocated by IEHMOVE.

```
//MOVEDS   JOB  09#550,GREEN
//         EXEC PGM=IEHMOVE
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   UNIT=disk,VOLUME=SER=333333,DISP=OLD
//DD1      DD   UNIT=disk,VOLUME=SER=111111,DISP=OLD
//DD2      DD   UNIT=(disk,,DEFER),DISP=OLD,
//              VOLUME=(PRIVATE,,SER=(222222))
//DD3      DD   VOLUME=(PRIVATE,RETAIN,SER=(444444)),
//              UNIT=disk,DISP=OLD
//SYSIN    DD   *
     MOVE   DSNAME=SEQSET1,TO=disk=222222,FROM=disk=444444
     MOVE   DSNAME=SEQSET2,TO=disk=222333,FROM=disk=444444
     MOVE   DSNAME=SEQSET3,TO=disk=222444,FROM=disk=444444
/*
```

The control statements are discussed below:

- SYSUT1 DD defines the disk device that is to contain the work data set.
- DD1 DD defines the system residence device.
- DD2 DD defines the mountable device on which the receiving volumes will be mounted as they are required.
- DD3 DD defines a mountable device on which the source volume is mounted. Because the RETAIN subparameter is included, the volume remains mounted until the job has completed.
- SYSIN DD defines the control data set, which follows in the input stream.
- MOVE moves the source data sets to volumes 222222, 222333, and 222444, respectively. The source data sets are scratched.

IEHMOVE EXAMPLE 2

In this example, three cataloged data sets are copied to a disk volume. Space is allocated by IEHMOVE. The catalog is not updated. The source data sets are not scratched.

```
//COPYPDS  JOB  09#550,GREEN
//         EXEC PGM=IEHMOVE
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   UNIT=disk,VOLUME=SER=222222,DISP=OLD
//DD1      DD   UNIT=disk,VOLUME=SER=111111,DISP=OLD
//DD2      DD   UNIT=disk,VOLUME=SER=222222,DISP=OLD
//DD3      DD   UNIT=disk,VOLUME=SER=333333,DISP=OLD
//SYSIN    DD   *
         COPY   DSNAME=SEQSET1,TO=disk=333333
         COPY   DSNAME=SEQSET3,TO=disk=333333
         COPY   DSNAME=SEQSET4,TO=disk=333333
/*
```

The control statements are discussed below:

- SYSUT1 DD defines the device that is to contain the work data set.
- DD1 DD defines the system residence device.
- DD2 DD defines a mountable device on which the source volume is mounted.
- DD3 DD defines a mountable device on which the receiving volume is mounted.
- SYSIN DD defines the control data set, which follows in the input stream.
- COPY copies the source data sets onto volume 333333.

IEHMOVE EXAMPLE 3

In this example, a partitioned data set (PARTSET1) is moved to a disk volume. In addition, a member (PARMEM3) from another partitioned data set (PARTSET2) is merged with the source members on the receiving volume. The source partitioned data set (PARTSET1) is scratched. Space is allocated by IEHMOVE.

```
//MOVEPDS  JOB  09#550,GREEN
//         EXEC PGM=IEHMOVE
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   UNIT=disk,VOLUME=SER=333000,DISP=OLD
//DD1      DD   UNIT=disk,VOLUME=SER=111111,DISP=OLD
//DD2      DD   UNIT=disk,VOLUME=SER=222111,DISP=OLD
//DD3      DD   UNIT=disk,VOLUME=SER=222222,DISP=OLD
//DD4      DD   UNIT=disk,VOLUME=SER=222333,DISP=OLD
//SYSIN    DD   *
     MOVE    PDS=PARTSET1,TO=disk=222333,FROM=disk=222111
  INCLUDE    DSNAME=PARTSET2,MEMBER=PARMEM3,FROM=disk=222222
/*
```

The control statements are discussed below:

- SYSUT1 DD defines the disk volume that is to contain the work data set.

- DD1 DD defines the system residence device.
- The DD2, DD3, and DD4 DD statements define mountable devices that are to contain the two source volumes and the receiving volume.
- SYSIN DD defines the control data set, which follows in the input stream.
- MOVE defines the source partitioned data set, the volume that contains it, and its receiving volume.
- INCLUDE includes a member from a second partitioned data set in the operation.

IEHMOVE EXAMPLE 4

In this example, a volume of data sets is moved to a disk volume. All data sets that are successfully moved are scratched from the source volume; however, any catalog entries pertaining to those data sets are not changed. Space is allocated by IEHMOVE. The work data set is deleted when the job step is completed.

```
//MOVEVOL  JOB  09#550,GREEN
//         EXEC PGM=IEHMOVE
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   UNIT=disk,VOLUME=SER=222222,DISP=OLD
//DD1      DD   UNIT=disk,VOLUME=SER=111111,DISP=OLD
//DD2      DD   UNIT=disk,VOLUME=SER=222222,DISP=OLD
//DD3      DD   UNIT=disk,VOLUME=SER=333333,DISP=OLD
//SYSIN    DD   *
         MOVE  VOLUME=disk=333333,TO=disk=222222,PASSWORD
/*
```

The control statements are discussed below:

- SYSUT1 DD defines the device that is to contain the work data set. The work data set is removed from the receiving volume when the job step is completed.
- DD1 DD defines the system residence device.
- DD2 DD defines the mountable device on which the receiving volume is mounted.
- DD3 DD defines a mountable device on which the source volume is mounted.
- SYSIN DD defines the control data set, which follows in the input stream.
- MOVE specifies a move operation for a volume of data sets and defines the source and receiving volumes. This statement also indicates that password-protected data sets are included in the operation.

IEHMOVE EXAMPLE 5

In this example, a partitioned data set is moved to a disk volume on which space has been previously allocated for the data set. The source data set is scratched. The work data set is deleted when the job step is completed.

```
//ALLOCATE JOB  09#550,GREEN
//         EXEC PGM=IEFBR14
//SET1     DD   DSNAME=PDSSET1,UNIT=disk,DISP=(NEW,KEEP),
//              VOLUME=SER=222222,SPACE=(TRK,(100,10,10)),
//              DCB=(RECFM=FB,LRECL=80,BLKSIZE=2000)
//         EXEC PGM=IEHMOVE
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   UNIT=disk,VOLUME=SER=222222,DISP=OLD
//DD1      DD   UNIT=disk,VOLUME=SER=111111,DISP=OLD
//DD2      DD   UNIT=disk,VOLUME=SER=222222,DISP=OLD
//DD3      DD   UNIT=disk,VOLUME=SER=333333,DISP=OLD
//SYSIN    DD   *
        MOVE   PDS=PDSSET1,TO=disk=222222,FROM=disk=333333
/*
```

The IEFBR14 job step is used to allocate space for data set PDSSET1 on a disk volume.

The control statements are discussed below:

- SYSUT1 DD defines the device that is to contain the work data set. The data set is removed at the completion of the program.
- DD1 DD defines the system residence device.
- DD2 DD defines the device on which the receiving volume is to be mounted.
- DD3 DD defines a mountable device on which the source volume is mounted.
- SYSIN DD defines the control data set, which follows in the input stream.
- MOVE specifies a move operation for the partitioned data set PDSSET1 and defines the source and receiving volumes.

IEHMOVE EXAMPLE 6

In this example, three partitioned data sets are moved from three separate source volumes to a disk volume. The source data set PDSSET3 is unloaded. (The record size exceeds the track capacity of the receiving volume.) The work data set is deleted when the job step is completed.

```
//ALLOCATE JOB  09#550,GREEN                                          72
//         EXEC PGM=IEFBR14
//SET1     DD   DSNAME=PDSSET1,UNIT=disk,DISP=(NEW,KEEP),
//             VOLUME=SER=222222,SPACE=(TRK,(50,10,5)),
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=1600)
//SET2     DD   DSNAME=PDSSET2,UNIT=disk,DISP=(NEW,KEEP),
//             VOLUME=SER=222222,SPACE=(TRK,(25,5,5)),
//             DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//SET3     DD   DSNAME=PDSSET3,UNIT=disk,DISP=(NEW,KEEP),
//             VOLUME=SER=222222,SPACE=(TRK,(25,5)),
//             DCB=(RECFM=U,BLKSIZE=5000)
//         EXEC PGM=IEHMOVE
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   UNIT=disk,VOLUME=SER=222222,DISP=OLD
//DD1      DD   UNIT=disk,VOLUME=SER=111111,DISP=OLD
//DD2      DD   UNIT=(disk,,DEFER),DISP=OLD,
//             VOLUME=(PRIVATE,,SER=(333333))
//DD3      DD   UNIT=disk,VOLUME=SER=222222,DISP=OLD
//SYSIN    DD   *
         MOVE   PDS=PDSSET1,TO=disk=222222,FROM=disk=333333
         MOVE   PDS=PDSSET2,TO=disk=222222,FROM=disk=222222
         MOVE   PDS=PDSSET3,TO=disk=222222,                          C
                FROM =disk=444444,UNLOAD
/*
```

The IEFBR14 job step is used to allocate space for the partitioned data sets PDSSET1, PDSSET2, and PDSSET3 on the receiving volume. The SPACE parameter in the SET3 DD statement allocates space for a sequential data set. This is necessary to successfully unload the partitioned data set PDSSET3.

For a discussion on estimating space allocations, refer to Data Management Services.

The DCB attributes of PDSSET3 are:

```
DCB=(RECFM=U,BLKSIZE=5000)
```

The unloaded attributes are:

```
DCB=(RECFM=FB,LRECL=80,BLKSIZE=800)
```

The control statements are discussed below:

- SYSUT1 DD defines the device that is to contain the work data set.
- DD1 DD defines the system residence device.
- DD2 DD defines a mountable device on which the source volumes are mounted as they are required.
- DD3 DD defines a mountable device on which the receiving volume is mounted.
- SYSIN DD defines the control data set, which follows in the input stream.
- MOVE specifies move operations for the partitioned data sets and defines the source and receiving volumes for each data set.

IEHMOVE EXAMPLE 7

In this example, a sequential data set is unloaded onto a 9-track, unlabeled tape volume (800 bits per inch). The work data set resides on the source volume and is deleted when the job step is completed.

```
                                                                      72
//UNLOAD   JOB  09#550,GREEN
//         EXEC PGM=IEHMOVE
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   UNIT=disk,VOLUME=SER=222222,DISP=OLD
//DD1      DD   UNIT=disk,VOLUME=SER=111111,DISP=OLD
//DD2      DD   UNIT=disk,VOLUME=SER=222222,DISP=OLD
//TAPEOUT  DD   UNIT=tape,VOLUME=SER=SCRTCH,DISP=OLD,
//              DCB=(DEN=2,RECFM=FB,LRECL=80,BLKSIZE=800),
//              LABEL=(,NL)
//SYSIN    DD   *
         MOVE   DSNAME=SEQSET1,TO=tape=SCRTCH,                       C
                FROM=disk=222222,TODD=TAPEOUT
/*
```

The control statements are discussed below:

- SYSUT1 DD defines the device that is to contain the work data set.
- DD1 DD defines the system residence device.
- DD2 DD defines a mountable device on which the source volume is mounted.
- TAPEOUT DD defines a mountable device on which the receiving tape volume is mounted. This statement also provides label and mode information.
- SYSIN DD defines the control data set, which follows in the input stream.
- MOVE moves the sequential data set SEQSET1 from a disk volume to the receiving tape volume. The data set is unloaded. The TODD parameter in this statement refers to the TAPEOUT DD statement for label and mode information.

IEHMOVE EXAMPLE 8

In this example, three unloaded sequential data sets are loaded from a labeled, 7-track tape volume (556 bits per inch) to a disk volume. Space is allocated by IEHMOVE. The example assumes that the disk volume is capable of supporting the data sets in their original forms.

```
                                                                      72
//LOAD     JOB  09#550,GREEN
//         EXEC PGM=IEHMOVE
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   UNIT=disk,VOLUME=SER=222222,DISP=OLD
//DD1      DD   UNIT=disk,VOLUME=SER=111111,DISP=OLD
//DD2      DD   UNIT=disk,VOLUME=SER=222222,DISP=OLD
//TAPESETS DD   UNIT=3420,
//              VOLUME=SER=001234,DISP=OLD,
//              LABEL=(1,SL),DCB=(DEN=1,TRTCH=C)
//SYSIN    DD   *
         MOVE   DSNAME=UNLDSET1,TO=disk=222222,                      C
                FROM=3420=(001234,1),FROMDD=TAPESETS
         MOVE   DSNAME=UNLDSET2,TO=disk=222222,                      C
                FROM=3420=(001234,2),FROMDD=TAPESETS
         MOVE   DSNAME=UNLDSET3,TO=disk=222222,                      C
                FROM=3420=(001234,3),FROMDD=TAPESETS
/*
```

The control statements are discussed below:

- SYSUT1 DD defines the device that is to contain the work data set.
- DD1 DD defines the system residence device.
- DD2 DD defines a mountable device on which the receiving volume is mounted.
- TAPESETS DD defines a mountable device on which the source tape volume is mounted. DCB information is provided in this statement.
- SYSIN DD defines the control data set, which follows in the input stream.
- MOVE moves the unloaded data sets to the receiving volume.

To move a data set from a tape volume that contains more than one data set, you must specify the sequence number of the data set in the list field of the FROM parameter on the utility control statement.

IEHMOVE EXAMPLE 9

In this example, two sequential data sets are copied from separate source volumes to a disk volume. Space is allocated by IEHMOVE. Only one 9-track tape unit is available for the operation.

```
                                                                      72
//DEFER    JOB  09#550,GREEN
//         EXEC PGM=IEHMOVE
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   UNIT=disk,VOLUME=SER=222222,DISP=OLD
//DD1      DD   UNIT=disk,VOLUME=SER=111111,DISP=OLD
//DD2      DD   UNIT=disk,VOLUME=SER=222222,DISP=OLD
//TAPE1    DD   VOLUME=SER=001234,UNIT=tape,DISP=OLD
//TAPE2    DD   VOLUME=SER=001235,UNIT=tape,DISP=OLD
//SYSIN    DD   *
         COPY   DSNAME=SEQSET1,TO=disk=222222,                       C
                FROM=3400=(001234,2),FROMDD=TAPE1
         COPY   DSNAME=SEQSET9,TO=disk=222222,                       C
                FROM=3400=(001235,4),FROMDD=TAPE2
/*
```

The control statements are discussed below:

- SYSUT1 DD defines the volume that is to contain the work data set.
- DD1 DD defines the system residence device.
- DD2 DD defines a mountable device on which the receiving volume is mounted.
- TAPE1 DD defines a mountable device on which the first volume to be processed is mounted. The source data set is the second data set on the volume.
- TAPE2 DD defines a mountable device on which the second volume to be processed is mounted when it is required. The source data set is the fourth data set on the volume.
- SYSIN DD defines the control data set, which follows in the input stream.

- COPY copies the second file of tape 001234 and the fourth file of tape 001235 to the receiving volume.

To copy a data set from a tape volume that contains more than one data set, you must specify the sequence number of the data set in the list field of the FROM parameter on the utility control statement.

IEHMOVE EXAMPLE 10

In this example, three unloaded partitioned data sets residing on an unlabeled tape volume mounted on device 282 are copied to a 3380 volume mounted on device 191.

```
                                                                      72
//LOAD     JOB  MEDDAUGH,PS40300439,MSGLEVEL=1
//          EXEC PGM=IEHMOVE
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   UNIT=191,VOLUME=SER=338000,DISP=OLD
//DD1      DD   UNIT=191,VOLUME=SER=338000,DISP=OLD
//TAPE1    DD   UNIT=282,VOLUME=SER=NLTAPE,DISP=OLD,
//              LABEL=(,NL),
//              DCB=(RECFM=FB,LRECL=80,BLKSIZE=800)
//SYSIN    DD   *
   COPY PDS=DSET1,FROM=282=(NLTAPE,1),                               X
        TO=191=338000,FROMDD=TAPE1
   COPY PDS=DSET2,FROM=282=(NLTAPE,2),                               X
        TO=191=338000,FROMDD=TAPE1
   COPY PDS=DSET3,FROM=282=(NLTAPE,3),                               X
        TO=191=338000,FROMDD=TAPE1
/*
```

The control statements are discussed below:

- SYSUT1 DD defines the work data set.
- DD1 DD defines the receiving volume.
- TAPE1 DD defines the source data sets. They are, in the order in which they reside on the volume, DSET1, DSET2, and DSET3.
- SYSIN DD defines the control data set, which follows in the input stream.
- COPY copies the unloaded partitioned data sets from the unlabeled tape to the receiving volume.

To copy data sets from an unlabeled tape, you must place a dummy label in the list field of the FROM parameter of the utility control statement. Following this dummy label, the sequence number of the data set must also be included in the same field. The unit address must appear in the device field of the FROM or TO parameter whenever you want to move from or copy to a specific device.

IEHMOVE EXAMPLE 11

In this example, the cataloged data set group A.B.C—which comprises data set A.B.C.X, A.B.C.Y, and A.B.C.Z—is moved from two disk volumes onto a third volume. Space is allocated by IEHMOVE. The catalog is updated to refer to the receiving volume. The source data sets are scratched.

```
//MOVEDSG  JOB  09#550,GREEN
//         EXEC PGM=IEHMOVE
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   UNIT=disk,VOLUME=SER=222222,DISP=OLD
//DD1      DD   UNIT=disk,VOLUME=SER=111111,DISP=OLD
//DD2      DD   UNIT=disk,VOLUME=SER=222222,DISP=OLD
//DD3      DD   UNIT=disk,VOLUME=SER=333333,DISP=OLD
//DD4      DD   UNIT=disk,VOLUME=SER=444444,DISP=OLD
//SYSIN    DD   *
         MOVE  DSGROUP=A.B.C,TO=disk=222222
/*
```

The control statements are discussed below:

- SYSUT1 DD defines the device that is to contain the work data set.
- DD1 DD defines the system residence device.
- DD2 DD defines a mountable device on which the receiving volume is mounted.
- DD3 DD defines a mountable device on which one of the source volumes is mounted.
- DD4 DD defines a mountable device on which one of the source volumes is mounted.
- SYSIN DD defines the control data set, which follows in the input stream.
- MOVE moves the specified data sets to volume 222222.

This example can be used to produce the same result without the use of the DD4 DD statement, using one less mountable disk device. With DD3 and DD4, both of the source volumes are mounted at the start of the job. With DD3 only, the 333333 volume is mounted at the start of the job. After the 333333 volume is processed, the utility requests that the operator mount the 444444 volume. In this case the DD3 statement is coded:

```
//DD3 DD UNIT=(disk,,DEFER),DISP=OLD,VOLUME=(PRIVATE,,
//       SER=(333333))
```

IEHMOVE EXAMPLE 12

In this example, the SYSCTLG data set is moved from a mountable disk volume to another mountable disk volume. Space is allocated by IEHMOVE. The source OS CVOL is scratched from the first disk volume.

```
//MOVECAT1 JOB  09#550,GREEN
//         EXEC PGM=IEHMOVE,PARM='POWER=3'
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   UNIT=disk,VOLUME=SER=333333,DISP=OLD
//DD1      DD   UNIT=disk,VOLUME=SER=111111,DISP=OLD
//DD2      DD   UNIT=disk,VOLUME=SER=222222,DISP=OLD
//SYSIN    DD   *
         MOVE  CATALOG,TO=disk=222222,CVOL=disk=111111
/*
```

The control statements are discussed below:

- SYSUT1 DD defines the device that is to contain the work data set.
- DD1 DD defines the mountable device on which the source volume is mounted.
- DD2 DD defines the mountable device on which the receiving volume is mounted.
- SYSIN DD defines the control data set, which follows in the input stream.
- MOVE specifies the move operation and defines the source and receiving volumes.

See "PARM Information on the EXEC Statement" on page 352 for a description of the POWER PARM.

IEHMOVE EXAMPLE 13

In this example, the OS CVOL entries for data set group A.B.C—which comprises the entries A.B.C.X, A.B.C.Y, and A.B.C.Z—are moved from a SYSCTLG data set to a mountable disk volume. If no OS CVOL exists on the receiving disk volume, one is created; if an OS CVOL does exist, the specified entries are merged into it. The last index of all entries in the source SYSCTLG is scratched. The work data set is deleted when the job step is completed.

```
//MOVECAT2 JOB  09#550,GREEN
//         EXEC PGM=IEHMOVE
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   UNIT=disk,VOLUME=SER=222222,DISP=OLD
//DD1      DD   UNIT=disk,VOLUME=SER=111111,DISP=OLD
//DD2      DD   UNIT=disk,VOLUME=SER=222222,DISP=OLD
//SYSIN    DD   *
         MOVE   CATALOG=A.B.C,TO=disk=222222,CVOL=disk=111111
/*
```

The control statements are discussed below:

- SYSUT1 DD defines the device that is to contain the work data set. (Because IEHMOVE deletes the work data set at the completion of the program, it can be contained on the receiving volume, provided there is space for it.)
- DD1 DD defines the mountable device on which the source volume is mounted.
- DD2 DD defines the mountable device on which the receiving volume is mounted.
- SYSIN DD defines the control data set, which follows in the input stream.
- MOVE specifies a move operation for selected entries and defines the source and receiving volumes.

IEHPROGM is a system utility used to modify system control data and to maintain data sets at an organizational level. IEHPROGM should only be used by those programmers locally authorized to do so.

IEHPROGM can be used to:

- Scratch a data set or a member.
- Rename a data set or a member.
- Catalog or remove catalog entries for a non-VSAM data set in an OS CVOL.
- Build or delete an index or alias in an OS CVOL (SYSCTLG data set).
- Connect or release two OS CVOLs.
- Build and maintain a generation data group index in an OS CVOL.
- Maintain data set passwords.

SCRATCHING A DATA SET OR MEMBER

IEHPROGM can be used to scratch the following from a DASD volume or volumes:

- Sequential, ISAM, partitioned, or BDAM data sets
- Members of a partitioned data set
- Password-protected data sets
- Data sets named by the operating system

A data set is considered scratched when its data set control block is removed from the volume table of contents (VTOC) of the volume on which it resides; its space is made available for reallocation.

A member is considered scratched when its name is removed from the directory of the partitioned data set in which it is contained. The space occupied by a scratched member is not available for reallocation until the partitioned data set is scratched or compressed. (When scratching a member of a partitioned data set, all aliases of that member should also be removed from the directory.)

If RACF is active, ALTER authorization is required to scratch a RACF-defined data set, and UPDATE authorization is required to scratch a member of a partitioned data set.

RENAMING A DATA SET OR MEMBER

IEHPROGM can be used to rename a data set or member that resides on a DASD volume. In addition, the program can be used to change any member aliases.

If RACF is active, ALTER authorization is required to rename a data set. UPDATE authorization is required to rename a member of a partitioned data set.

CATALOGING A DATA SET IN AN OS CVOL

IEHPROGM can be used to catalog a non-VSAM sequential, ISAM, partitioned, or BDAM data set in an OS CVOL. The program catalogs a data set by generating an entry, containing the data set name and associated volume information, in the index of the OS CVOL. A valid TTR pointer is not placed in the DSCB until the first time the data set is referenced.

The catalog function is used to catalog a non-VSAM data set in an OS CVOL that was not cataloged when it was created.

IEHPROGM can also delete OS CVOL entries for a non-VSAM data set by removing the data set name and associated volume information from the OS CVOL.

The cataloging function of IEHPROGM differs from a DISP=(,CATLG) specification in a DD statement in that the DISP=(,CATLG) specification cannot catalog a data set on a volume other than the system residence volume unless the system residence volume is properly connected to the other volume. (See "Connecting or Releasing Two OS CVOLs" on page 406.)

The "uncataloging" function of IEHPROGM differs from a DISP=(...,UNCATLG) specification in a DD statement in that the DISP=(...,UNCATLG) specification cannot remove an entry from the SYSCTLG data set on a volume other than the system residence volume unless the two volumes are properly connected.

You should **not** use the IEHPROGM CATLG/UNCATLG functions in place of DISP=(,CATLG) or DISP=(,UNCATLG) in a multi-step job. If a data set is to be "uncataloged" during termination of a step, use DISP=(OLD,UNCATLG).

BUILDING OR DELETING AN INDEX IN AN OS CVOL

IEHPROGM can be used to build a new index in an OS CVOL or to delete an existing index. In building an index, the program automatically creates as many higher level indexes as are necessary to complete the specified structure.

IEHPROGM can be used to delete one or more indexes from an index structure; however, an index cannot be deleted if it contains any entries. That is, it cannot be deleted if it refers to a lower level index or if it is part of a structure indicating the fully qualified name of an OS CVOL cataloged data set.

Figure 138 on page 406 shows an index structure before and after a build operation. The left portion of the figure shows two data sets cataloged in an OS CVOL, A.Y.YY and A.B.X.XX, before the build operation. The right-hand portion of the figure shows the index structure after the build operation, which was used to build index A.B.C.D.E. Note in the left portion of the figure that index levels C and D do not exist before the build operation. These levels are automatically created when the level E index is built.

When the level E index is subsequently deleted, the level C and D indexes are not automatically deleted by the program. To delete these index levels, delete: A.B.C.D.E, A.B.C.D, and A.B.C, in that order. The level B index cannot be deleted because data set A.B.X.XX and the X level index are dependent upon the level B index.

BUILDING OR DELETING AN INDEX ALIAS IN AN OS CVOL

IEHPROGM can be used to assign an alternative name (alias) to the highest level index of an OS CVOL or to delete an OS CVOL index alias previously assigned. An alias cannot, however, be assigned to the highest level of a generation data group index.

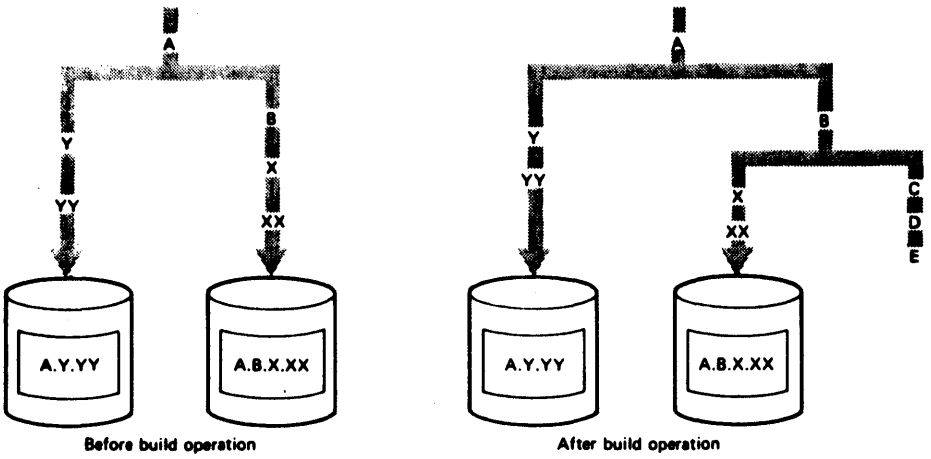


Figure 138. Index Structure Before and After an IEHPROGM Build Operation

Figure 139 on page 406 shows an alias, XX, that is assigned to index A (a high level index). The cataloged data set A.B.C can be referred to as either A.B.C or XX.B.C.

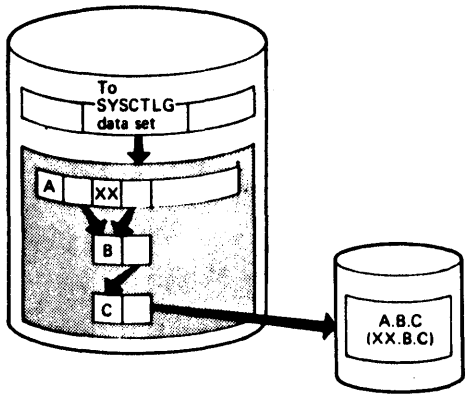


Figure 139. Building an Index Alias Using IEHPROGM

CONNECTING OR RELEASING TWO OS CVOLS

IEHPROGM can be used to connect an OS CVOL to a second OS CVOL by placing an entry into a high level index on the first OS CVOL. The entry contains an index name and the volume serial number and device type of the second OS CVOL. The program can subsequently release the OS CVOLs by removing the entry from the high level index. If two OS CVOLs are connected:

- The SYSCTLG data set must be created on the second volume for cataloging of data sets having the same high level index as the connected index.

- A high level index can only be connected to one second OS CVOL, but chaining is possible from a second to a third OS CVOL, etc.

Before any OS CVOL can be accessed by the system, it must be defined in the ICF or VSAM master catalog. For details on how this is done, see Catalog Users Guide.

Figure 140 shows how one OS CVOL can be connected to a second OS CVOL. Any subsequent index search for index X on the first control volume is carried to the second control volume.

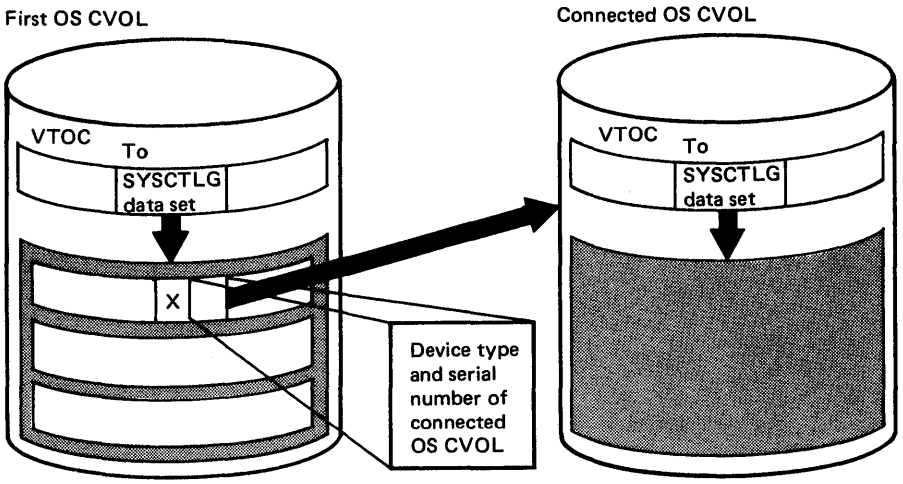


Figure 140. Connecting an OS CVOL to a Second OS CVOL Using IEHPROGM

The index name of each high level index existing on the second OS CVOL must be present in the first OS CVOL; when a new high level index is placed on a second OS CVOL, the first OS CVOL should be connected to the second OS CVOL.

Figure 141 on page 408 shows three OS CVOLs connected to one OS CVOL. All volumes are accessible through high level indexes X, Y, and Z.

BUILDING AND MAINTAINING A GENERATION DATA GROUP INDEX IN AN OS CVOL

IEHPROGM can be used to build an index structure in an OS CVOL for a generation data group and to define what action should be taken when the index overflows.

The lowest level index in the structure can contain up to 255 entries for successive generations of a data set. If the index overflows, the oldest entry is removed from the index, unless otherwise specified (in which case all entries are removed). If desired, the program can be used to scratch all generation data sets whose entries are removed from the index.

Figure 142 on page 408 shows the index structure created for generation data group A.B.C. In this example, provision is made for up to five subsequent entries in the lowest level index.

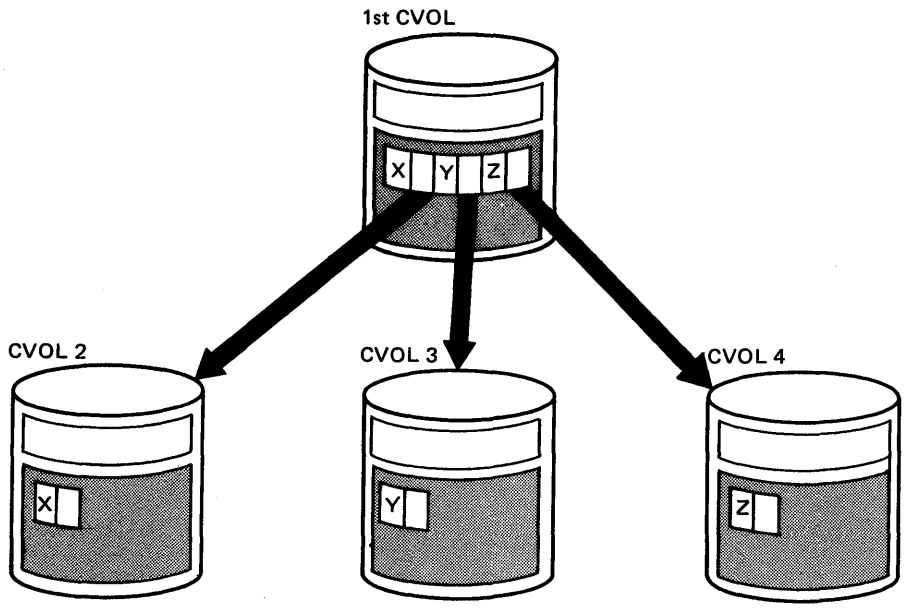


Figure 141. Connecting Three OS CVOLs Using IEHPROGM

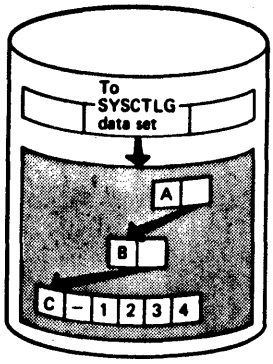


Figure 142. Building a Generation Data Group Index Using IEHPROGM

Before a generation data group can be cataloged as such on an OS CVOL, a generation data group index must exist. Otherwise, a generation data set is cataloged as an individual data set, rather than as a generation.

When creating and cataloging a generation data set on an OS CVOL, the user must provide the necessary DCB information. See Data Management Services for a discussion of how DCB attributes are provided for a generation data group.

MAINTAINING DATA SET PASSWORDS

IEHPROGM can be used to maintain non-VSAM password entries in the PASSWORD data set and to alter the protection status of BDAM data sets in the data set control block (DSCB). For a complete description of data set passwords and the PASSWORD data set, see System Programming Library: Data Management and Data Management Services.

A data set can have one of three types of password protection, as indicated in the DSCB for BDAM data sets and in the tape label for tape data sets. (See Debugging Handbook for the format of the DSCB. See Magnetic Tape Labels and File Structure for a description of tape labels.)

The possible types of data set password protection are:

- No protection, which means that no passwords are required to read or write the data set.
- Read/write protection, which means that a password is required to read or write the data set.
- Read-without-password protection, which means that a password is required only to write the data set; the data set can be read without a password.

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

A data set can have one or more passwords assigned to it; each password has an entry in the PASSWORD data set. A password assigned to a data set can allow read and write access, or only read access to the data set.

Figure 143 on page 410 shows the relationship between the protection status of data set ABC and the type of access allowed by the passwords assigned to the data set. Passwords ABLE and BAKER are assigned to data set ABC. If no password protection is set in the DSCB or tape label, data set ABC can be read or written without a password. If read/write protection is set in the DSCB or tape label, data set ABC can be read with either password ABLE or BAKER and can be written with password ABLE. If read-without-password protection is set in the DSCB or tape label, data set ABC can be read without a password and can be written with password ABLE; password BAKER is never needed.

Before IEHPROGM is used to maintain data set passwords, the PASSWORD data set must reside on the system residence volume. IEHPROGM can then 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 from an entry in the PASSWORD data set.

Each entry in the PASSWORD data set contains the name of the protected data set, the password, the protection mode of the password, an access counter, and 77 bytes of optional user data. The protection mode of the password defines the type of access allowed by the password and whether the password is a control password or secondary password. The initial password, added to the PASSWORD data set for a particular data set, is marked in the entry as the control password for that data set. The second and subsequent passwords added for the same data set are marked as secondary passwords.

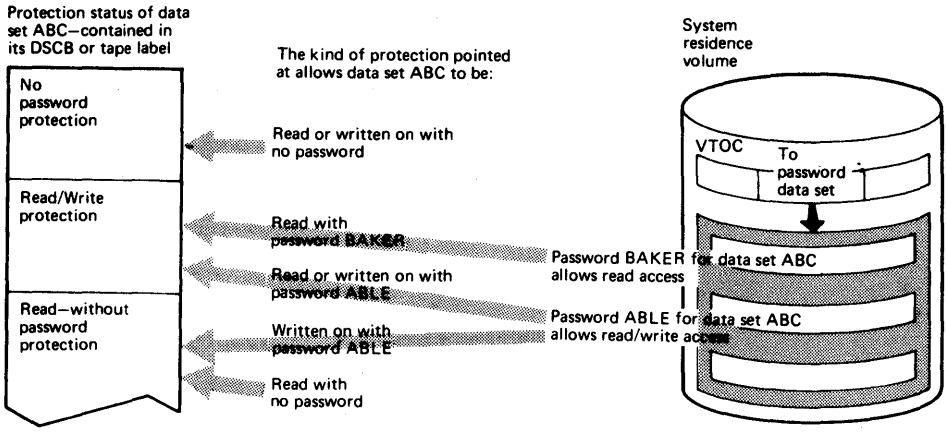


Figure 143. Relationship between the Protection Status of a Data Set and Its Passwords

For BDAM data sets, IEHPROGM updates the protection status in the DSCB when a control password entry is added, replaced, or deleted. This permits setting and resetting the protection status of an existing BDAM data set at the same time its passwords are added, replaced, or deleted. IEHPROGM automatically alters the protection status of a data set in the DSCB if the following conditions are met:

- The control password for the data set is being added, replaced, or deleted.
- The data set is online.
- The volume on which the data set resides is specified on the utility control statement, or the data set is cataloged.
- The data set is not allocated within the IEHPROGM job.

For tape data sets, IEHPROGM cannot update the protection status in the tape label when a password entry is added, replaced, or deleted. Protection status in a tape label must be set with JCL.

Passwords to be added, replaced, deleted, or listed can be specified on utility control statements or can be entered by the console operator. IEHPROGM issues a message to the console operator when a password on a utility control statement is either missing or invalid. The message contains the job name, step name, and utility control statement name and identifies the particular password that is missing or invalid. Two invalid passwords are allowed per password entry on each utility control statement before the request is ignored; a total of five invalid passwords is allowed for the password entries on all the utility control statements in a job step before the step is canceled.

Adding Data Set Passwords

When a password is added for a data set, an entry is created in the PASSWORD data set with the specified data set name, password name, protection mode of the password (read/write or read only), and the optional 77 characters of user-supplied data. The access counter in the entry is set to zero.

The control password for a data set must always be specified to add, replace, or delete secondary passwords. The control password should not be specified, however, to list information from a secondary password entry.

Secondary passwords can be assigned to a data set to restrict some users to reading the data set or to record the number of times certain users access the data set. The access counter in each password entry provides a count of the number of times the password was used to successfully open the data set.

If a control password for a BDAM, online data set is added, the protection status of the data set (read/write or read-without-password) is set in the DSCB. However, the data set to be protected must not be allocated within the same job as the one in which IEHPROGM is executed. If it is allocated, the DSCB cannot be accessed and the protection status is not set. If the data set to be protected is being created within the same job, use JCL to set the protection status in the DSCB.

Replacing Data Set Passwords

Any of the following information may be replaced in a password entry: the password, protection mode (read/write or read only) of the password, and the 77 characters of user data. The protection status of a data set can be changed by replacing the control entry for the data set.

If the control entry of a BDAM, online data set is replaced, the DSCB is also reset to indicate any change in the protection status of the data set. Therefore, you should ensure that the volume is online when changing the protection status of a BDAM data set.

Deleting Data Set Passwords

When a control password entry is deleted from the PASSWORD data set, all secondary password entries for that data set are also deleted. However, when a secondary entry is deleted, no other password entries are deleted.

If the control password entry is deleted for an online, BDAM data set, the protection status of the data set in the DSCB is also changed to indicate no protection. When deleting a control password for a BDAM data set, the user should ensure that the volume is online. If the volume is not online, the password entry is removed, but data set protection is still indicated in the DSCB; the data set cannot be accessed unless another password is added for that data set.

If the control password entry is deleted for a tape data set, the tape volume cannot be accessed unless another password is added for that data set.

The delete function should be used to delete all the password entries for a scratched data set to make the space available for new entries.

Listing Password Entries

A list of information from any entry in the PASSWORD data set can be obtained in the SYSPRINT data set by providing the password for that entry. The list includes: the number of times the password has been used to successfully open the data set; the type of password (control password or secondary password) and type of access allowed by the password (read/write or read-only); and the user data in the entry. Figure 144 on page 412 shows a sample list of information printed from a password entry.

```
DECIMAL ACCESS COUNT= 000025
PROTECT MODE BYTE= SECONDARY, READ ONLY
USER DATA FIELD= ASSIGNED TO J. BROWN
```

Figure 144. Listing of a Password Entry

INPUT AND OUTPUT

IEHPROGM uses the following input:

- One or more data sets containing system control data to be modified.
- A control data set that contains utility control statements used to control the functions of the program.

IEHPROGM produces the following output:

- A modified object data set or volume(s).
- A message data set that contains error messages and information from the PASSWORD data set.

RETURN CODES

IEHPROGM returns a code in register 15 to indicate the results of program execution. The return codes and their meanings are listed below.

Codes	Meaning
00 (00 hex)	Successful completion.
04 (04)	A syntax error was found in the name field of the control statement or in the PARM field in the EXEC statement. Processing continues.
08 (08)	A request for a specific operation was ignored because of an invalid control statement or an otherwise invalid request. The operation is not performed.
12 (0C)	An input/output error was detected when trying to read from or write to SYSPRINT, SYSIN or the VTOC. The job step is terminated.
16 (10)	An unrecoverable error exists. The job step is terminated.

Figure 145. IEHPROGM Return Codes

CONTROL

IEHPROGM is controlled by job control statements and utility control statements.

Job control statements are used to:

- Execute or invoke the program.
- Define the control data set.

- Define volumes and/or devices to be used during the course of program execution.
- Prevent data sets from being deleted inadvertently.
- Prevent volumes from being demounted before they have been completely processed by the program.
- Suppress listing of utility control statements.

Utility control statements are used to control the functions of the program and to define those data sets or volumes that are to be modified.

JOB CONTROL STATEMENTS

Figure 146 on page 414 shows the job control statements for IEHPROGM.

With the exception of the SYSIN and SYSPRINT DD statements, all DD statements in Figure 146 on page 414 are used as device allocation statements, rather than as true data definition statements. Because IEHPROGM modifies the internal control blocks created by device allocation DD statements, the DSNAME parameter, if supplied, will be ignored by IEHPROGM. (All data sets are defined explicitly or implicitly by utility control statements.)

Note: Unpredictable results may occur in multitasking environments where dynamic allocation/deallocation of devices, by other tasks, causes changes in the TIOT during IEHPROGM execution.

PARM Information on the EXEC Statement

Additional information can be specified in the PARM parameter of the EXEC statement to control the number of lines per page on the output listing and to suppress printing of utility control statements. The EXEC statement can be coded:

```
//EXEC   PGM=IEHPROGM[,PARM=[LINECNT=xx,]
                         [PRINT|NOPRINT]]
```

The LINECNT parameter specifies the number of lines per page in the listing of the SYSPRINT data set; xx is a 2-digit number, from 01 through 99. If LINECNT is omitted, or if an error is encountered in the LINECNT parameter, the number of lines per page will be 45.

The PRINT value specifies that the utility control statements are to be written to the SYSPRINT data set. If neither PRINT nor NOPRINT is coded, PRINT is assumed.

The NOPRINT value specifies that utility control statements are not to be written to the SYSPRINT data set. Suppressing printing of utility control statements assures that passwords assigned to data sets remain confidential. However, suppressing printing may make it difficult to interpret error messages because the relevant utility control statement is not printed before the message.

Statement	Use
JOB	Initiates the job.
EXEC	Specifies the program name (PGM=IEHPROGM) or, if the job control statements reside in a procedure library, the procedure name. Additional PARM information can be specified to control the number of lines per page on the output listing and to suppress printing of utility control statements. See "PARM Information on the EXEC Statement."
SYSPRINT DD	Defines a sequential message data set.
anyname1 DD	Defines a permanently mounted volume. (The system residence volume is considered to be a permanently mounted volume.)
anyname2 DD	Defines a mountable device type.
SYSIN DD	Defines the control data set. The control data set normally follows the job control statements in the input stream; however, it can be defined as a member of a procedure library.

Figure 146. IEHPROGM Job Control Statements

SYSPRINT DD Statement

The block size for the SYSPRINT data set must be a multiple of 121. Any blocking factor can be specified.

anyname1 DD Statement

One anyname1 DD statement must be included for each permanently mounted volume referred to in the job step.

The anyname1 DD statement can be entered:

```
//anyname1 DD UNIT=xxxx,VOLUME=SER=xxxxxx,DISP=OLD
```

The UNIT and VOLUME parameters define the device type and volume serial number. The DISP=OLD specification prevents the inadvertent deletion of a data set. (The anyname1 DD statement is arbitrarily assigned the ddname DD1 in the IEHPROGM examples.)

anyname2 DD Statement

One anyname2 DD statement must be included for each mountable device to be used in the job step. Multiple statements may be coded as long as each anyname is unique.

The anyname2 DD statement can be coded in the following ways:

```
//anyname2 DD  VOLUME=SER=xxxxxx,UNIT=xxxx,DISP=OLD

//anyname2 DD  VOLUME=(PRIVATE,SER=xxxxxx),
//    UNIT=(xxxx,,DEFER),DISP=OLD
```

The second example can be used to specify deferred mounting when a large number of magnetic tapes or DASD volumes are to be processed in one application of the program.

The UNIT and VOLUME parameters define the device type and volume serial number. The DISP=OLD specification prevents the

inadvertent deletion of a data set. Unit affinity cannot be used on DD statements defining mountable devices. (The anyname2 DD statement is arbitrarily assigned the ddname DD2 in the IEHPROGM examples.)

When IEHPROGM is dynamically invoked in a job step containing a program other than IEHPROGM, the DD statements defining mountable devices must be included in the job stream prior to DD statements defining data sets required by the other program.

Refer to Appendix B, "DD Statements for Defining Mountable Devices" on page 443 for instructions on defining mountable volumes.

SYSIN DD Statement

The block size for the SYSIN data set must be a multiple of 80. Any blocking factor can be specified.

UTILITY CONTROL STATEMENTS

Figure 147 on page 416 shows the utility control statements for IEHPROGM.

Continuation requirements for utility control statements are described in "Continuing Utility Control Statements" on page 5. Note that continued lines need not begin in column 16 for IEHPROGM.

SCRATCH Statement

The SCRATCH statement is used to scratch a data set or member from a DASD volume. A data set or member is scratched only from the volume(s) designated in the SCRATCH statement. This function does not delete OS CVOL entries for scratched data sets.

A SCRATCH operation will not be executed if the data set or volume is being used by a program executing concurrently. "DISP=OLD" on the DD statement only prevents the inadvertent deletion of a data set. It does not ensure exclusive use of the data set during execution of the job step.

For multivolume data sets, all volumes specified must be online.

The format of the SCRATCH statement is:

[label]	SCRATCH	{VTOC|DSNAME=name} ,VOL=device=(list) [,PURGE] [,MEMBER=name] [,SYS]

RENAME Statement

The RENAME statement is used to change the true name or alias of a data set or member residing on a DASD volume. The name is changed only on the designated volume(s). The rename operation does not update the OS CVOL.

A RENAME operation will not be executed if the data set or volume is being used by a program executing concurrently.

Statement	Use
SCRATCH	Scratches a data set or a member from a DASD volume.
RENAME	Changes the name or alias of a data set or member residing on a DASD volume.
CATLG	Generates an entry in the index of an OS CVOL.
UNCATLG	Removes an entry from the lowest level index of an OS CVOL.
BLDX	Creates a new index in the OS CVOL (SYSCTLG data set).
DLTX	Removes a low level index from an OS CVOL.
BLDA	Assigns an alias to an index at the highest level of an OS CVOL.
DLTA	Deletes an alias previously assigned to an index at the highest level of an OS CVOL.
CONNECT	Connects two OS CVOLs together using a high level index name.
RELEASE	Removes a high level index name from one OS CVOL that served as a connector or pointer to a second OS CVOL.
BLDG	Builds an index in an OS CVOL for a generation data group and defines what action should be taken when the index overflows.
ADD	Adds a password entry in the PASSWORD data set.
REPLACE	Replaces information in a password entry.
DELETEP	Deletes an entry in the PASSWORD data set.
LIST	Formats and lists information from a password entry.

Figure 147. IEHPROGM Utility Control Statements

For multivolume data sets, all volumes specified must be online.

If you do not code the MEMBER parameter, then the entire data set is renamed.

The format of the RENAME statement is:

[label]	RENAME	DSNAME=name ,VOL=device=(list) ,NEWNAME=name [,MEMBER=name]

CATLG Statement

The CATLG statement is used to generate a non-VSAM entry in the index of an OS CVOL. If additional levels of indexes are required in the OS CVOL, this function automatically creates them.

When cataloging generation data sets and the index becomes full, see "BLDG (Build Generation Data Group Index) Statement" on page 419 for the action to be taken.

To catalog VSAM data sets in an ICF or VSAM catalog, see Catalog Users Guide and Access Method Services Reference.

When device is represented by a group name (for example, SYSDA) instead of a generic name (for example, 3350 or 3400) in the VOL parameter, the catalog operation does not enter the device type code in the OS CVOL. Instead, it places a unique entry in the device type field of the OS CVOL. The allocation of the device for this entry may not be satisfactory to the user. The generic name should be used if the group name was generated for one or more device types. When the system is subsequently generated, this entry may no longer be valid; that is, entries for all such group names should be deleted and then the group names should be recataloged after a subsequent generation of the system.

When cataloging data sets residing on tape, specify the volume serial number and the data set sequence number as follows:

VOL=device**=(**serial**,**seqno**,...)**

If a data set is created on a 9-track dual density tape unit (3400-6), the data set can be cataloged with a device specification of 3400-3 for an 1600 bits per inch tape or 3400-5 for a 6250 bits per inch tape. If a device specification of 3400-6 is made when the data set is cataloged, any subsequent retrieval of that data set is made on a dual density unit.

The format of the CATLG statement is:

[label]	**CATLG**	**DSNAME=**name **,VOL=**device**=(**list**)|(**serial**,**seqno**)** **[,CVOL=**device**=**serial**]**

UNCATLG Statement

The UNCATLG statement is used to remove a non-VSAM entry from the index of the OS CVOL. If the entry removed was the last entry in the index, that index and all higher, unneeded, indexes, with the exception of the highest-level index, are removed from the OS CVOL.

The format of the UNCATLG statement is:

[label]	**UNCATLG**	**DSNAME=**name **[,CVOL=**device**=**serial**]**

BLDX (Build Index) Statement

The BLDX statement is used to create a new index in an OS CVOL. If the creation of an index requires that higher level indexes be created, this function automatically creates them.

The format of the BLDX statement is:

[label]	BLDX	INDEX=name [,CVOL=device=serial]

DLTX (Delete Index) Statement

The DLTX statement is used to remove an index from an OS CVOL. Only an index that has no entries can be removed.

Because this function does not delete higher level indexes, it must be used repetitively to delete an entire structure. For example, to delete a generation data group index structure A.B.C.names, you must code the following sequence of statements:

```
DLTX INDEX=A.B.C

DLTX INDEX=A.B

DLTX INDEX=A
```

The format of the DLTX statement is:

[label]	DLTX	INDEX=name [,CVOL=device=serial]

BLDA (Build Index Alias) Statement

The BLDA statement is used to assign an alias to an index at the highest level of an OS CVOL.

The format of the BLDA statement is:

[label]	BLDA	INDEX=name ,ALIAS=name [,CVOL=device=serial]

DLTA (Delete Index Alias) Statement

The DLTA statement is used to delete an alias previously assigned to an index at the highest level of an OS CVOL.

The format of the DLTA statement is:

[label]	DLTA	ALIAS=name [,CVOL=device=serial]

CONNECT Statement

The CONNECT statement is used to place an entry in the high level index of an OS CVOL. The entry identifies a second OS CVOL by its device type and volume serial number. In addition, it contains an index name identifying the index to be searched for (during subsequent index searches) on the second OS CVOL.

This function does not create an index on the second OS CVOL.

The CONNECT statement does not create a SYSCTLG data set on the connected control volume. Before cataloging the first data set on a connected control volume, the user must define a SYSCTLG data set on that volume. This can be done with the following DD statement:

```
//ddname DD DSNAME=SYSCTLG,UNIT=xxxx,DISP=(,KEEP),
//          SPACE=(CYL,1),VOLUME=SER=xxxxxx
```

If a job requires an auxiliary control volume to complete a catalog search, the user need not have the auxiliary control volume mounted before the job is begun. (The user does not have to remember the volume on which a particular data set is cataloged.) The system directs the operator to mount an auxiliary control volume if it is needed.

Before any OS CVOL can be accessed by the system, it must be defined in the ICF or VSAM master catalog. For details, see Catalog Users Guide.

The format of the CONNECT statement is:

[label]	**CONNECT**	**INDEX=**name **,VOL=**device**=**serial **[,CVOL=**device**=**serial**]**

RELEASE (Disconnect) Statement

The RELEASE statement is used to remove an entry from the high level index of an OS CVOL. This disconnects, in effect, a second OS CVOL from the first OS CVOL. The RELEASE statement does not delete an index from the second OS CVOL.

The format of the RELEASE statement is:

[label]	**RELEASE**	**INDEX=**name **[,CVOL=**device**=**serial**]**

BLDG (Build Generation Data Group Index) Statement

The BLDG statement is used to build an index for a generation data group, and to define what action should be taken when the index overflows.

To delete a generation data group index structure, use the "DLTX (Delete Index) Statement" on page 418.

The format of the BLDG statement is:

[label]	BUILD	INDEX=name ,ENTRIES=n [,CVOL=device=serial] [,EMPTY] [,DELETE]

ADD (Add a Password) Statement

The ADD statement is used to add a password entry in the PASSWORD data set. When the control entry for a BDAM, online data set is added, the indicated protection status of the data set is set in the DSCB; when a secondary entry is added, the protection status in the DSCB is not changed.

The format of the ADD statement is:

[label]	ADD	DSNAME=name [,PASWORD2=new-password] [,CPASWORD=control-password] [,TYPE=code] [,VOL=device=(list)] [,DATA='user-data']

REPLACE (Replace a Password) Statement

The REPLACE statement is used to replace any or all of the following information in a password entry: the password name, protection mode (read/write or read only) of the password, and user data. When the control entry for a BDAM, online data set is replaced, the protection status of the data set is changed in the DSCB if necessary; when a secondary entry is replaced, the protection status in the DSCB is not changed.

The format of the REPLACE statement is:

[label]	REPLACE	DSNAME=name [,PASWORD1=current-password] [,PASWORD2=new-password] [,CPASWORD=control-password] [,TYPE=code] [,VOL=device=(list)] [,DATA='user-data']

DELETEP (Delete a Password) Statement

The DELETEP statement is used to delete an entry in the PASSWORD data set. If a control entry is deleted, all the secondary entries for that data set are also deleted. If a secondary entry is deleted, only that entry is deleted. When the control entry for a BDAM, online data set is deleted, the protection status in the DSCB is set to indicate that the data set is no longer protected.

The format of the DELETEP statement is:

[label]	DELETEP	DSNAME=name [,PASWORD1=current-password] [,CPASWORD=control-password] [,VOL=device=(list)]

LIST (List Information from a Password) Statement

The LIST statement is used to format and print information from a password entry.

The format of the LIST statement is:

[label]	LIST	DSNAME=name ,PASWORD1=current-password

Parameters	Applicable Control Statements	Description of Parameters
ALIAS	BLDA DLTA	**ALIAS=name** specifies an unqualified name to be assigned as the alias or to be deleted from the index. The name must not exceed 8 characters.
CPASWORD	ADD	**CPASWORD=control-password** specifies the control password for the data set. CPASWORD must be specified unless this is the first password assigned to the data set, in which case PASWORD2 specifies the password to be added.
	DELETEP REPLACE	**CPASWORD=control-password** CPASWORD must be specified unless the control entry is being changed or deleted, in which case PASWORD1 specifies the control password.

Parameters	Applicable Control Statements	Description of Parameters
CVOL	CATLG UNCATLG BLDX DLTX BLDA DLTA CONNECT RELEASE BLDG	**CVOL=**device=serial For CATLG, UNCATLG, BLDX, DLTX and BLDG, CVOL specifies the OS CVOL on which the search for the index (entry, for UNCATLG) is to begin. For BLDA and DLTA, CVOL specifies the OS CVOL on which the entry is to be made or deleted. For CONNECT and RELEASE, CVOL specifies specifies the device type and volume serial number of the first OS CVOL. **If CVOL is omitted:** For CATLG and UNCATLG, the search begins with the ICF or VSAM master/JOBCAT/STEPCAT catalog. For BLDX, DLTX, BLDA, DLTA, CONNECT, RELEASE and BLDG, the system attempts to locate the proper (the first, for CONNECT) OS CVOL by checking the ICF or VSAM master catalog for an OS CVOL pointer alias name equal to the high level index specified in the INDEX (ALIAS, for DLTA) parameter. The OS CVOL must be defined in the ICF or VSAM master catalog as: SYSCTLG.Vserial, where serial must equal the serial number of the CVOL. See Catalog Users Guide for more information. **Default:** The search begins with the ICF or VSAM master catalog (or JOBCAT/STEPCAT, if specified).
DATA	ADD REPLACE	**DATA='**user-data**'** specifies the user data to be placed in the password entry. The user data has a maximum length of 77 bytes and must be enclosed in apostrophes. Any other apostrophes contained within the user data must be entered as two single apostrophes. If DATA is omitted from an ADD operation, 77 blanks are used. If DATA is omitted from a REPLACE operation, current user data is not changed.
DELETE	BLDG	**DELETE** specifies that generation data sets are scratched after their entries are removed from the index.

Parameters	Applicable Control Statements	Description of Parameters
DSNAME	SCRATCH RENAME CATLG UNCATLG ADD REPLACE DELETEP LIST	**DSNAME=**name specifies the fully qualified name of the data set to be either scratched or renamed; the fully qualified name of the partitioned data set that contains the member to be scratched or renamed; the fully qualified name of the data set to be cataloged or uncataloged; or the fully qualified name of the data set whose password entry is to be added, replaced, deleted, or listed. The qualified name must not exceed 44 characters, including delimiters.
EMPTY	BLDG	**EMPTY** specifies that all entries be removed from the generation data group index when it overflows. This deletes all index entries for all of the generation data sets. **Default:** The entries with the largest generation numbers will be maintained in the catalog when the generation data group index overflows.
ENTRIES	BLDG	**ENTRIES=**n specifies the number of entries to be contained in the generation data group index; n must not exceed 255.
INDEX	BLDG	**INDEX=**name specifies the 1- to 35-character qualified name of the generation data group index.
	BLDX DLTX	**INDEX=**name specifies the qualified name of the index to be created or deleted. The qualified name must not exceed 44 characters, including delimiters.
	BLDA	**INDEX=**name specifies the unqualified name of the index to which an alias name is to be assigned. The unqualified name must not exceed 8 characters.
	CONNECT RELEASE	**INDEX=**name specifies the unqualified index name to be entered or removed from the high level index on the first OS CVOL. The unqualified name must not exceed 8 characters.

Parameters	Applicable Control Statements	Description of Parameters
MEMBER	SCRATCH RENAME	**MEMBER=**name specifies a member name or alias of a member (in the named data set) to be renamed or removed from the directory of a partitioned data set. This name is not validity-checked because all members must be accessible, whether the name is valid or not. **Default:** The entire data set or volume of data sets specified by name is changed or scratched.
NEWNAME	RENAME	**NEWNAME=**name specifies the new fully qualified name or alias name for the data set or the new member.
PASWORD1	REPLACE DELETEP LIST	**PASWORD1=**current-password specifies the password in the entry to be listed, changed, or deleted. **Default:** The operator is prompted for the current password.
PASWORD2	ADD REPLACE	**PASWORD2=**new-password specifies the new password to be added or assigned to the entry. If the password is not to be changed, the current password must also be specified as the new password. The password can consist of 1 to 8 alphameric characters. **Default:** The operator is prompted for a new password.
PURGE	SCRATCH	**PURGE** specifies that each data set specified by DSNAME or VTOC be scratched, even if its expiration date has not elapsed. **Default:** The specified data sets are scratched only if their expiration dates have elapsed.
SYS	SCRATCH	**SYS** specifies that data sets which have not expired but which have names that begin with "AAAAAAAA.AAAAAAAA.AAAAAAAA.AAAAAAAA." or "SYSnnnnn.T" and "F," "V," or "A" in position 19 are scratched. These are names assigned to data sets by the operating system. This parameter is valid only when VTOC is specified. If the name of the data set to be scratched begins with SYS, nnnnn is the date in dddyy format.

Parameters	Applicable Control Statements	Description of Parameters
TYPE	ADD REPLACE	**TYPE=**<u>code</u> specifies the protection code of the password and, if a control password entry is to be changed for or assigned to a BDAM, online data set, specifies the protection status of the data set. The values that can be specified for <u>code</u> are: **1** specifies that the password is to allow both read and write access to the data set; if a control password is being assigned or changed, read/write protection is set in the DSCB. **2** specifies that the password is to allow only read access to the data set; if control password is being assigned or changed, read/write protection is set in the DSCB. **3** specifies that the password is to allow both read and write access to the data set; if a control password is being assigned or changed, read-without-password protection is set in the DSCB. **Default:** For ADD, if this parameter is omitted, the new password is assigned the same protection code as the control password for the data set. If a control password is being "added," TYPE=3 is the default. For REPLACE, the protection is not changed.

Parameters	Applicable Control Statements	Description of Parameters
VOL	CONNECT	**VOL=**device=serial specifies the device type and serial number of the second OS CVOL. This information is placed in the high level index of the first OS CVOL.
	ADD REPLACE DELETEP SCRATCH RENAME	**VOL=**device=(list) specifies the device type and serial number(s) of the volume(s), limited to 50, that contain the data set(s). If only one serial number is listed in list, it need not be enclosed in parentheses. For ADD, REPLACE and DELETEP, if omitted, the protection status in the DSCB is not set or changed, unless the data set is cataloged in the OS CVOL. This parameter is not necessary for secondary password entries, or if the desired protection status in the DSCB is already set or is not to be changed by ADD or REPLACE. For SCRATCH and RENAME, if VTOC or MEMBER is specified, VOL cannot specify more than one volume. Caution should be used when specifying VTOC if VOL specifies the system residence volume.
	CATLG	**VOL=**device=(list)|(serial,seqno) specifies the device type, serial numbers, and data set sequence numbers (for tape volumes) of the volumes (up to 50) that contain the data sets to be cataloged in the OS CVOL. The volume serial numbers must appear in the same order in which they were originally encountered (in DD statements within the input stream) when the data set was created. seqno is valid only for data sets which reside on tape.
VTOC	SCRATCH	**VTOC** specifies that all data sets on the specified volume, except those protected by a password or those whose expiration dates have not expired, are scratched. Password-protected data sets are scratched if the correct password is provided. The effect of VTOC is modified when it is used with PURGE or SYS.

IEHPROGM EXAMPLES

The following examples illustrate some of the uses of IEHPROGM. Figure 148 can be used as a quick-reference guide to IEHPROGM examples. The numbers in the "Example" column point to the examples that follow.

Operation	Mount Volumes	Comments	Example
SCRATCH	Disk	VTOC is scratched.	1
SCRATCH UNCATLG	Disk	Two data sets are scratched and their entries removed from the OS CVOL.	2
RENAME, UNCATLG CATLG	Disks	A data set is renamed on two mountable devices; the old data set name is removed from the OS CVOL. The data set is cataloged under its new name.	3
UNCATLG	Disk	Index structures for three generation data sets are deleted from the OS CVOL.	4
RENAME DELETEP, and ADD	Disk	A data set is renamed. The old passwords are deleted and new passwords are assigned.	5
LIST and REPLACE	Disk	A password entry is listed. Protection mode and status are changed, and user data is added.	6
RENAME	Disk	A member of a partitioned data set is renamed.	7
CATLG and CONNECT	Disk	One OS CVOL is connected to another.	8
BLDG, RENAME and CATLG	Disk	A generation data group index is built, three data sets are renamed and entered in the index.	9
BLDG	Disk	A new generation data group index is built and updated through JCL. A model DSCB is created. New generations are added.	10

Figure 148. IEHPROGM Example Directory

Examples that use **disk** or **tape** in place of actual device numbers must be changed before use. See "DASD and Tape Device Support" on page 3 for valid device number notation.

IEHPROGM EXAMPLE 1

In the following example, all data sets are scratched from the volume table of contents of a mountable volume. Because the system residence volume is not referred to, no DD1 DD statement is necessary in the job stream.

```
//SCRVTOC  JOB  09#550,BROWN
//         EXEC PGM=IEHPROGM
//SYSPRINT DD   SYSOUT=A
//DD2      DD   UNIT=disk,VOLUME=SER=222222,DISP=OLD
//SYSIN    DD   *
      SCRATCH   VTOC,VOL=disk=222222, SYS
/*
```

The SCRATCH statement, used in this example, indicates that all data sets (including those system data sets beginning with AAAAAAAA.AAAAAAAA.AAAAAAAA.AAAAAAAA) whose expiration dates have expired are scratched from the specified volume.

IEHPROGM EXAMPLE 2

In this example, two data sets are scratched: SET1 is scratched on volume 222222, and A.B.C.D.E is scratched on volume 222222. Both data sets are uncataloged.

```
//SCRDSETS JOB  09#550,BROWN
//         EXEC PGM=IEHPROGM
//SYSPRINT DD   SYSOUT=A
//DD1      DD   UNIT=disk,VOLUME=SER=111111,DISP=OLD
//DD2      DD   UNIT=disk,DISP=OLD,VOLUME=SER=222222
//SYSIN    DD   *
      SCRATCH   DSNAME=SET1,VOL=disk=222222
      UNCATLG   DSNAME=SET1
      SCRATCH   DSNAME=A.B.C.D.E,VOL=disk=222222
      UNCATLG   DSNAME=A.B.C.D.E
/*
```

The utility control statements are discussed below:

- The first SCRATCH statement specifies that SET1, which resides on volume 222222, is scratched.
- The first UNCATLG statement specifies that SET1 is uncataloged.
- The second SCRATCH statement specifies that A.B.C.D.E, which resides on volume 222222, is scratched.
- The second UNCATLG statement specifies that A.B.C.D.E is uncataloged.

IEHPROGM EXAMPLE 3

In this example, the name of a data set is changed on two mountable volumes. The old data set name is removed from the OS CVOL and the data set is cataloged under its new data set name.

```
                                                                      72
//RENAMEDS JOB  09#550,BROWN
//         EXEC PGM=IEHPROGM
//SYSPRINT DD   SYSOUT=A
//DD1      DD   VOLUME=SER=111111,UNIT=disk,DISP=OLD
//DD2      DD   UNIT=(disk,,DEFER),DISP=OLD,
//              VOLUME=(PRIVATE,SER=(222222,333333))
//SYSIN    DD   *
   RENAME  DSNAME=A.B.C,NEWNAME=NEWSET,                              C
           VOL=disk=(222222,333333)
  UNCATLG  DSNAME=A.B.C
    CATLG  DSNAME=NEWSET,VOL=disk=(222222,333333)
/*
```

The control statements are discussed below:

- RENAME specifies that data set A.B.C, which resides on volumes 222222 and 333333, is renamed NEWSET.
- UNCATLG specifies that data set A.B.C is uncataloged.
- CATLG specifies that NEWSET, which resides on volumes 222222 and 333333, is cataloged in the OS CVOL.

IEHPROGM EXAMPLE 4

In this example, three data sets—A.B.C.D.E.F.SET1, A.B.C.G.H.SET2, and A.B.I.J.K.SET3—are uncataloged.

```
//DLTSTRUC JOB  09#550,BROWN
//         EXEC PGM=IEHPROGM
//SYSPRINT DD   SYSOUT=A
//DD1      DD   UNIT=disk,VOLUME=SER=111111,DISP=OLD
//SYSIN    DD   *
      UNCATLG   DSNAME=A.B.C.D.E.F.SET1
      UNCATLG   DSNAME=A.B.C.G.H.SET2
      UNCATLG   DSNAME=A.B.I.J.K.SET3
/*
```

The control statements are discussed below:

- The UNCATLG statements specify that data sets A.B.C.D.E.F.SET1, A.B.C.G.H.SET2, and A.B.I.J.K.SET3 are uncataloged.

IEHPROGM EXAMPLE 5

In this example, a data set is renamed. The data set passwords assigned to the old data set name are deleted. Then two passwords are assigned to the new data set name.

If the data set is not cataloged, a message is issued indicating that the LOCATE macro instruction failed.

```
                                                                      72
//ADDPASS  JOB  09#550,BROWN
//         EXEC PGM=IEHPROGM,PARM='NOPRINT'
//SYSPRINT DD   SYSOUT=A
//DD1      DD   VOLUME=(PRIVATE,SER=222222),DISP=OLD,
//              UNIT=(disk,,DEFER)
//SYSIN    DD   *
  RENAME DSNAME=OLD,VOL=disk=222222,NEWNAME=NEW
 DELETEP DSNAME=OLD,PASWORD1=KEY
     ADD DSNAME=NEW,PASWORD2=KEY,TYPE=1,                              C
               DATA='SECONDARY IS READ'
     ADD DSNAME=NEW,PASWORD2=READ,CPASWORD=KEY,TYPE=2,                C
               DATA='ASSIGNED TO J. DOE'
/*
```

The utility control statements are discussed below:

- RENAME specifies that the data set called OLD is renamed NEW. The operator is required to supply a password to rename the old data set.
- DELETEP specifies that the entry for the password KEY is deleted. Because KEY is a control password in this example, all the password entries for the data set name are deleted. The VOL parameter is not needed because the protection status of the data set as set in the DSCB is not to be changed; read/write protection is presently set in the DSCB, and read/write protection is desired when the passwords are reassigned under the new data set name.
- The ADD statements specify that entries are added for passwords KEY and READ. KEY becomes the control password and allows both read and write access to the data set. READ

becomes a secondary password and allows only read access to the data set. The VOL parameter is not needed, because the protection status of the data set is still set in the DSCB.

IEHPROGM EXAMPLE 6

In this example, information from a password entry is listed. Then the protection mode of the password, the protection status of the data set, and the user data are changed.

```
                                                                      72
//REPLPASS JOB  09#550,BROWN
         EXEC PGM=IEHPROGM,PARM='NOPRINT'
//SYSPRINT DD   SYSOUT=A
//DD1      DD   UNIT=disk,VOLUME=SER=111111,DISP=OLD
//DD2      DD   VOLUME=(PRIVATE,SER=(222222,333333)),
//              UNIT=(disk,,DEFER),DISP=OLD
//SYSIN    DD   *
    LIST   DSNAME=A.B.C,PASWORD1=ABLE
  REPLACE  DSNAME=A.B.C,PASWORD1=ABLE,                                 C
              PASWORD2=ABLE,TYPE=3,                                    C
              VOL=disk=(222222,333333),                                C
              DATA='NO SECONDARIES; ASSIGNED TO DEPT 31'
/*
```

The utility control statements are discussed below:

- LIST specifies that the access counter, protection mode, and user data from the entry for password ABLE are listed. Listing the entry permits the content of the access counter to be recorded before the counter is reset to zero by the REPLACE statement.
- REPLACE specifies that the protection mode of password ABLE is to be changed to allow both read and write access and that the protection status of the data set is changed to write-only protection. The VOL parameter is required because the protection status of the data set is changed and the data set, in this example, is not cataloged. Because this is a control password, the CPASWORD parameter is not required.

IEHPROGM EXAMPLE 7

In this example, a member of a partitioned data set is renamed.

```
                                                                      72
//REN      JOB  09#550,BROWN
//         EXEC PGM=IEHPROGM
//SYSPRINT DD   SYSOUT=A
//DD1      DD   VOL=SER=222222,DISP=OLD,UNIT=disk
//SYSIN    DD *
 RENAME    VOL=disk=222222,DSNAME=DATASET,NEWNAME=BC,                 C
               MEMBER=ABC
/*
```

The control statements are discussed below:

- DD1 DD defines a permanently mounted volume.
- SYSIN DD defines the input data set, which follows in the input stream.

- RENAME specifies that member ABC in the partitioned data set DATASET, which resides on a disk volume, is renamed BC.

IEHPROGM EXAMPLE 8

In this example, a new OS CVOL (SYSCTLG data set) is defined and connected to an existing OS CVOL. A data set is then cataloged in the new OS CVOL.

```
//LNKX      JOB
//STEP1     EXEC PGM=IEHPROGM
//SYSPRINT  DD   SYSOUT=A
//NEWCVOL   DD   DSN=SYSCTLG,UNIT=disk,VOL=SER=222222,
//               DISP=(,KEEP),SPACE=(TRK,(10,1))
//DD1       DD   UNIT=disk,VOL=SER=111111,DISP=SHR
//SYSIN     DD   *
   CATLG DSNAME=SYSCTLG.V222222,VOL=disk=222222
   CONNECT INDEX=AA,VOL=disk=222222
   CATLG DSNAME=AA.BB,VOL=disk=PACK14
/*
```

This example assumes that the OS CVOL on volume 111111 was previously defined in the ICF or VSAM master catalog with an OS CVOL pointer, and "AA" was defined in the ICF or VSAM master catalog as an alias of the OS CVOL pointer. See Catalog Users Guide for details on how this is done.

The utility control statements are discussed below:

- NEWCVOL DD allocates space for the new OS CVOL.
- The first CATLG statement establishes an OS CVOL pointer in the ICF or VSAM master catalog for the new OS CVOL.
- The CONNECT statement causes the new OS CVOL (on volume 222222) to be connected to the old OS CVOL (on volume 111111), such that any catalog management requests coming to the old OS CVOL having a high level index name of AA will be routed to the new OS CVOL.
- The second CATLG statement will cause the data set AA.BB to be cataloged in the new OS CVOL on volume 222222. Since this is the first request to update the new OS CVOL, this will cause the new OS CVOL to be formatted before the catalog entry is made.

IEHPROGM EXAMPLE 9

In this example, a generation data group index for generation data group A.B.C is built in an OS CVOL. Three existing noncataloged, nongeneration data sets are renamed; the renamed data sets are entered as generations in the generation data group index.

```
                                                                      72
//BLDINDEX JOB
//         EXEC PGM=IEHPROGM
//SYSPRINT DD   SYSOUT=A
//DD1      DD   UNIT=disk,VOLUME=SER=111111,DISP=OLD
//DD2      DD   UNIT=(disk,,DEFER),DISP=OLD,
//              VOLUME=(PRIVATE,,SER=(222222))
//SYSIN    DD   *
  BLDG     INDEX=A.B.C,ENTRIES=10,CVOL=disk=111111
  RENAME   DSNAME=DATASET1,VOL=disk=222222,                          C
               NEWNAME=A.B.C.G0001V00
  RENAME   DSNAME=DATASET2,VOL=disk=222222,                          C
               NEWNAME=A.B.C.G0002V00
  RENAME   DSNAME=DATASET3,VOL=disk=222222,                          C
               NEWNAME=A.B.C.G0003V00
  CATLG DSNAME=A.B.C.G0001V00,VOL=disk=222222,                       C
        CVOL=disk=111111
  CATLG DSNAME=A.B.C.G0002V00,VOL=disk=222222,                       C
        CVOL=disk=111111
  CATLG DSNAME=A.B.C.G0003V00,VOL=disk-222222,                       C
        CVOL=disk=111111
/*
```

The control statements are discussed below:

- DD1 DD defines the volume on which the SYSCTLG data set resides.
- BLDG specifies the generation group name A.B.C and makes provision for ten entries in the index. The oldest generation is uncataloged when the index becomes full. No generations are scratched.
- The RENAME statements rename three nongeneration data sets residing on a disk volume.
- The CATLG statements enter the renamed data sets in the generation data group index and catalog them in the OS CVOL.

Because the DCB parameters were supplied when the nongeneration data sets were created, no DCB parameters are now specified; therefore, no model DSCB is required. See Example 10 for information on how to create a model DSCB.

IEHPROGM EXAMPLE 10

In this example, an IEHPROGM job step, STEPA, creates a model DSCB and builds a generation data group index. STEP B, an IEBGENER job step, creates and catalogs a sequential generation data set from data in the input stream. STEP C, an IEBGENER job step, creates and catalogs a second generation with new DCB attributes.

This example assumes that the OS CVOL with serial number 111111 was previously defined in the ICF or VSAM master catalog with an OS CVOL pointer, and "A" was defined in the ICF or VSAM master catalog as an alias of the OS CVOL pointer. See Catalog Users Guide for details on how this is done.

```
//BLDINDX  JOB
//STEPA    EXEC PGM=IEHPROGM
//SYSPRINT DD   SYSOUT=A
//BLDDSCB  DD   DSNAME=A.B.C,DISP=(,KEEP),SPACE=(TRK,(0)),
//              DCB=(LRECL=80,RECFM=FB,BLKSIZE=800),
//              VOLUME=SER=111111,UNIT=disk
//SYSIN    DD   *
          BLDG INDEX=A.B.C,ENTRIES=10,EMPTY,DELETE
/*
//STEPB    EXEC PGM=IEBGENER
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   DUMMY
//SYSUT2   DD   DSNAME=A.B.C(+1),UNIT=disk,DISP=(,CATLG),
//              VOLUME=SER=222222,SPACE=(TRK,20)
//SYSUT1   DD   DATA

(input data)

//STEPC    EXEC PGM=IEBGENER
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   DUMMY
//SYSUT2   DD   DSNAME=A.B.C(+1),UNIT=disk,DISP=(,CATLG),
//              DCB=(LRECL=80,RECFM=FB,BLKSIZE=1600),
//              VOLUME=SER=222222,SPACE=(TRK,20)
//SYSUT1   DD   DATA

(input data)

/*
```

The control statements are discussed below:

STEPA:

- BLDDSCB DD creates a model DSCB on the OS CVOL volume.
- SYSIN DD indicates that the control data set follows in the input stream.
- BLDG specifies the generation data group name A.B.C and makes provision for ten entries in the group. When the index is filled, it is emptied, and all of the generations are deleted.

STEPB:

- SYSUT2 DD defines an output sequential generation data set. The generation data set is assigned the absolute generation and version number G0001V00 in the index.
- SYSUT1 DD defines the input data set, which follows in the input stream.

STEPC:

- SYSUT2 DD defines a second output sequential generation data set. The generation data set is assigned the absolute generation and version number G0002V00 in the index. The specified DCB attributes override those initially specified in the model DSCB. The DCB attributes specified when the model DSCB was created remain unchanged; that is, those attributes are applicable when you catalog a succeeding generation unless you specify overriding attributes at that time.
- SYSUT1 defines the input data set, which follows in the job stream.

Any subsequent job that causes the deletion of the generations should include DD statements defining the devices on which the volumes containing those generations are to be mounted. The OS CVOL entry is deleted for each generation for which no DD statement is included at that time, but the generation itself is not deleted.

After the generation data group is emptied, the new generations continue to be assigned generation numbers according to the last generation number assigned before the empty operation. To reset the numbering operation (that is, to reset to G0000V00 or G0001V00), it is necessary to delete the catalog entries for all the old generation data sets and then rename and recatalog, beginning with G0000V00.

IFHSTATR PROGRAM

IFHSTATR is a system utility used to format and print information from type 21 SMF (system management facilities) records, which provide error statistics by volume (ESV) data.

Figure 149 shows the format of the type 21 record.

	Bytes of Record Descriptor Word			
0	System Indicator	Record Type	Time of Day	
4	Time of Day (continued)		Current Date	
8	Current Date (continued)		System Identification	
12	System Identifier		Length of rest of record including this field	
16	Volume Serial Number			
20	Volume Serial Number (continued)		22 Channel Unit Address	
24	UCB Type			
28	Temporary Read Errors	Temporary Write Errors	Start I/O's	
32	Permanent Read Errors	Permanent Write Errors	Noise Blocks	Erase Gaps
36	Erase Gaps (continued)	Cleaner Actions		Tape Density
40	Block Size		Reserved	

Figure 149. Type 21 SMF Record Format with ESV Data

Error statistics by volume (ESV) records are retrieved from the IFASMFDP tape or from SYS1.MAN (on tape).

ASSESSING THE QUALITY OF TAPES IN A LIBRARY

The statistics gathered by SMF in type 21 records can be very useful in assessing the quality of tapes in a library. IFHSTATR prints type 21 records in the same order that they were gathered, that is, date/time sequence. You may find it useful to sort type 21 records into volume serial number sequence, into channel unit sequence, and into error occurrence sequence to aid in analyzing the condition of tapes in the library.

The IFHSTATR report helps to identify deteriorating media (tapes); occasionally poor performance from a particular tape drive can also be identified. The permanent read error counter or permanent write error counter is increased by one each time the tape error recovery routines (ERPs) determine that the error is permanent and is returned to the user with indication of a permanent I/O error. If a SYNAD routine to handle such errors is present, the counts in these fields can be greater than one. The temporary read error counter and temporary write error counter are increased when the ERP initially handles an error condition corrected in the ERP. The severity of a temporary error can be estimated by analyzing either the erase gap counter for write errors or the noise block and cleaner action counters

for read errors. The erase gap counter is increased each time a write error is retried.

For example, if the temporary write error counter contains 2 and the erase gap counter contains 5, the ERP was entered twice for write error recovery. The average recovery actions were 2.5 per error (actually may have been 1 and 4). The cleaner action counter is only increased every fourth read retry. A ratio of one cleaner action to one temporary read error indicates, in general, recovery on the fifth retry (the first retry after the cleaner action). A ratio of 10 cleaner actions to one temporary error indicates that recovery is, in general, a result of reading the tape in the opposite direction (reading backward on a read forward tape or reading forward on a read backward tape). The noise block counter is increased once for each noise record (record less than minimum read length) encountered.

In analyzing IFHSTATR reports, the usage (SIO) count should also be considered, because it is the count of all start I/Os to the tape drive, except those issued by the ERP in the course of error recovery. The usage count can be used to determine the ratio of error-free accesses of the tape to total accesses of the tape.

INPUT AND OUTPUT

IFHSTATR uses as input type 21 SMF (system management facilities) records from an IFASMFDP tape, which contain information about errors on magnetic tape. IFHSTATR processes only type 21 records; if none are found, a message is written to the output data set.

IFHSTATR produces as output an output data set, which contains information selected from type 21 records. The output takes the form of 121-byte unblocked records, with an American National Standards Institute (ANSI) control character in the first byte of each record.

Figure 150 shows a sample of printed output from IFHSTATR.

VOLUME SERIAL	DATE	CPU ID	MOD NO	TIME OF DAY	CHANNEL / UNIT	TEMP READ	TEMP WRITE	PERM READ	PERM WRITE	NOISE BLOCKS	ERASE GAPS	CLEANER ACTIONS	USAGE (SIO'S)	TAPE DENSITY	BLOCK LENGTH
001021	82/309	BB	40	15:55:07	181	1	0	0	0	1	0	0	10	0800	80
001022	82/309	AA	40	15:56:02	184	10	0	0	0	0	0	0	28	1600	121
000595	82/309	CC	50	15:56:20	283	0	10	0	0	0	10	0	28	0800	50

Figure 150. Sample Output from IFHSTATR

CONTROL

IFHSTATR is controlled by job control statements only. Utility control statements are not used.

JOB CONTROL STATEMENTS

Figure 151 on page 437 shows the job control statements for IFHSTATR.

Statement	Use
JOB	Initiates the job.
EXEC	Specifies the program name (PGM=IFHSTATR).
SYSUT1 DD	Defines the input data set and the device on which it resides. The DSNAME, UNIT, VOLUME, LABEL, DCB, and DISP parameters should be included if the data set is not copied from another program.
SYSUT2 DD	Defines the sequential data set on which the output is to be written.

Figure 151. IFHSTATR Job Control Statements

The output data set can reside on any output device supported by BSAM.

The LRECL and BLKSIZE parameters are not specified by IFHSTATR. This information is taken from the DCB parameter on the SYSUT1 DD statement or from the tape label.

IFHSTATR EXAMPLE

This example shows the JCL needed to produce a report.

```
//REPORT   JOB
//         EXEC PGM=IFHSTATR
//SYSUT1   DD   DSNAME=COPY.SMFDATA,DISP=SHR
//SYSUT2   DD   SYSOUT=A
/*
```

Note: COPY.SMFDATA is a copy of the SYS1.MANx data set made with the IFASMFDP utility program.

APPENDIX A. EXIT ROUTINE LINKAGE

Utility programs can be linked to user-supplied exit routines for additional processing.

LINKING TO AN EXIT ROUTINE

Linking to an exit routine from a utility program is accomplished by the utility program in one of the following ways:

- If the exit routine is for label processing or totaling, or if the exit routine is specified in the IEBTCRIN program by OUTREC or ERROR, linkage is performed by the BALR instruction.
- In all other cases, linkage is performed by using the LINK macro instruction.

The LINK macro instruction contains the symbolic name of the entry point of an exit routine and, if required, a list of parameters.

For further information on the use of the LINK macro instruction, see "LINK or ATTACH Macro Instruction" on page 13.

At the time of the linkage operation:

- General register 1 contains the starting address of the parameter list.
- General register 13 contains the address of the register save area. This save area must not be used by user label processing routines. See Appendix C, "Processing User Labels" on page 446.
- General register 14 contains the address of the return point in the utility program.
- General register 15 contains the address of the entry point to the exit routine.

Registers 1 through 14 must be restored before control is returned to the utility program.

The exit routine must be contained in either the job library or the link library.

The parameter lists passed to label processing routines and parameter lists passed to nonlabel processing routines are described in the topics that follow.

LABEL PROCESSING ROUTINE PARAMETERS

The parameters passed to a user's label processing routine are addresses of: the 80-byte label buffer, the DCB being processed, the status information if an uncorrectable input/output error occurs, and the totaling area.

The 80-byte label buffer contains an image of the user label when an input label is being processed. When an output label is being processed, the buffer contains no significant information at entry to the user's label processing routine. When the utility program has been requested to generate labels, the user's label processing routine must construct a label in the label buffer.

If standard user labels (SUL) are specified on the DD statement for a data set, but the data set has no user labels, the system still takes the specified exits to the appropriate user's routine. In such a case, the user's input label processing routine is entered with the buffer address parameter set to zero.

The format and content of the DCB are presented in Data Management Macro Instructions.

Bit 0 of flag 1 in the DCB-address parameter is set to a value of 0 except when:

- Volume trailer or header labels are being processed at volume switch time.
- The trailer labels of a MOD data set are being processed (when the data set is opened).

If an uncorrectable input/output error occurs while reading or writing a user label, the appropriate label processing routine is entered with bit 0 of flag 2 in the status information address parameter set on. The three low order bytes of this parameter contain the address of standard status information as supplied for SYNAD routines. (The SYNAD routine is not entered.)

NONLABEL PROCESSING ROUTINE PARAMETERS

Figure 152 shows the programs from which exits can be taken to nonlabel processing routines, the names of the exits, and the parameters available for each exit routine.

Program	Exit	Parameters
IEBGENER	KEY	Address at which key is to be placed (record follows key); address of DCB.
	DATA	Address of SYSUT1 record; address of DCB.
	IOERROR	Address of DECB; cause of the error and address of DCB. (Address in lower order three bytes and cause of error in high order byte.)
IEBCOMPR	ERROR	Address of DCB for SYSUT1; address of DCB for SYSUT2.[1]
	PRECOMP	Address of SYSUT1 record; length of SYSUT1 record, address of SYSUT2 record; length of SYSUT2 record.
IEBPTPCH	INREC	Address of input record; length of the input record.
	OUTREC	Address of output record; length of the output record.
IEBTCRIN	ERROR	Address of the error record; address of a fullword which contains the record length.
	OUTREC	Address of the normal record; address of a fullword which contains the record length.

Figure 152. Parameter Lists for Nonlabel Processing Exit Routines

Note to Figure 152 on page 439:

[1] The IOBAD pointer in the DCB points to a location that contains the address of the corresponding data event control block (DECB) for these records. The format of the DECB is illustrated as part of the BSAM READ macro instruction in Data Management Macro Instructions.

RETURNING FROM AN EXIT ROUTINE

An exit routine returns control to the utility program by means of the RETURN macro instruction in the exit routine.

The format of the RETURN macro instruction is:

[label]	RETURN	[(r,r)] [,RC=n|(15)]

where:

(r,r)
specifies the range of registers, from 0 to 15, to be reloaded by the utility program from the register save area. For example, (14,12) indicates that all registers except register 13 are to be restored. If this parameter is omitted, the registers are considered properly restored by the exit routine.

RC=
specifies a decimal return code in register 15. If RC is omitted, register 15 is loaded as specified by (r,r).

RC values can be coded:

n
specifies a return code to be placed in the 12 low order bits of register 15.

(15)
specifies that general register 15 already contains a valid return code.

The user's label processing routine must return a code in register 15 as shown in Figure 153 unless:

- The buffer address was set to zero before entry to the label processing routine. In this case, the system resumes normal processing regardless of the return code.
- The user's label processing routine was entered after an uncorrectable output error occurred. In this case the system attempts to resume normal processing.

Figure 153 shows the return codes that can be issued to utility programs by user exit routines. Slightly different return codes are used for the UPDATE=INPLACE option of the IEBUPDTE program. See "IEBUPDTE Program" on page 293.

For a list of return codes issued by IEBTCRIN at job termination, see "IEBTCRIN Program" on page 267.

Type of Exit	Return Code	Action
Input Header or Trailer Label	0	The system resumes normal processing. If there are more labels in the label group, they are ignored.
	4	The next user label is read into the label buffer area and control is returned to the user's routine. If there are no more labels, normal processing is resumed.
	16	The utility program is terminated on request of the user routine.
Output Header or Trailer Label	0	The system resumes normal processing. No label is written from the label buffer area.
	4	The user label is written from the label buffer area. The system then resumes normal processing.
	8	The user label is written from the label buffer area. If fewer than eight labels have been created, the user's routine again receives control so that it can create another user label. If eight labels have been created, the system resumes normal processing.
	16	The utility program is terminated on request of the user routine.
Totaling Exits	0	Processing continues, but no further exits are taken.
	4	Normal operation continues.
	8	Processing ceases, except for EOD processing on output data set (user label processing).
	16	Utility program is terminated.
All other exits (except IEBPTPCH's exit OUTREC and IEBTCRIN's exits ERROR and OUTREC)	0-11 (Set to next multiple of four)	Return code is compared to highest previous return code; the higher is saved and the other discarded. At the normal end of job, the highest return code is passed to the calling processor.
	12 or 16	Utility program is terminated and this return code is passed to the calling processor.
ERROR	0	Record is not placed in the error data set. Processing continues with the next record.
	4	Record is placed in the error data set (SYSUT3).
	8	Record is not placed in error data set but is processed as a valid record (sent to OUTREC and SYSUT2 if specified). IEBTCRIN removes the EDW from an edited MTDI record before processing continues.
	16	Utility program is terminated.

Figure 153 (Part 1 of 2). Return Codes That Must Be Issued by User Exit Routines

Type of Exit	Return Code	Action
OUTREC (IEBPTPCH)	4	Record is not placed in normal output data set.
	12 or 16	Utility program is terminated.
	Any other number	Record is placed in normal output data set (SYSUT2).
OUTREC (IEBTCRIN)	0	Record is not placed in normal output data set.
	4	Record is placed in normal output data set (SYSUT2).
	16	Utility program is terminated.

Figure 153 (Part 2 of 2). Return Codes That Must Be Issued by User Exit Routines

Further information on the use of the RETURN macro instruction is contained in Supervisor Services and Macro Instructions.

APPENDIX B. DD STATEMENTS FOR DEFINING MOUNTABLE DEVICES

When defining mountable devices to be used by system utility programs IEHPROGM, IEHMOVE, or IEHLIST, the user must consider the implications of the DD statements used to define those devices.

DD statement parameters must ensure that no one else has access to either the volume or the data set. In any case, caution should be used when altering volumes that are permanently resident or reserved.

Under normal conditions, a mountable device should not be shared with another job step; that is, if a utility program is used to update a volume on a mountable device, the volume being updated must remain mounted until the operation is completed.

Following are ways to ensure that mountable devices are not shared:

- Specify DEFER in a DD statement defining a mountable device.
- Specify a volume count in the VOLUME parameter of a DD statement that is greater than the number of mountable devices to be allocated.
- Specify PRIVATE in a DD statement defining a mountable device.

For a detailed discussion, see the publication JCL.

DD STATEMENT EXAMPLES

In the following examples of DD statements, an IBM DASD is indicated as the mountable device. Alternative parameters are stacked.

Examples that use **disk** in place of actual device numbers must be changed before use. See "DASD and Tape Device Support" on page 3 for valid device number notation.

DD EXAMPLE 1

This DD statement makes a specific request for a private, nonsharable volume or volumes to be mounted on a single device.

```
//DD1  DD  UNIT=(disk,,DEFER),DISP=(,KEEP),
//         VOLUME=(PRIVATE,SER=(123456)),
//         SPACE=(CYL,(1,1))
```

A utility program causes a mount message to be issued for a specific volume when the volume is required for processing by the program. The user should supply the operator with the clearly marked volume or volumes to be mounted during the job step.

This DD statement ensures that the volume integrity of a mountable volume is maintained. If only one volume is to be processed, it is mounted at the start of the job step and demounted at the end of the step. If additional volumes are processed, they are mounted and demounted when needed by the utility program. The last volume to be processed is demounted at the end of the job step.

DD EXAMPLE 2

This DD statement makes a request for a private, nonsharable volume.

```
//DD2  DD  UNIT=(disk,,DEFER),VOLUME=PRIVATE,DISP=(NEW,KEEP),
//         SPACE=(CYL,(1,1))
```

The results of this statement are identical to those shown in DD Example 1.

If a specific unit is requested and the volume serial number is not given in the DD statement, the user must be certain that either: (1) the desired volume is already mounted on that unit, or (2) a volume is not mounted, causing the system to issue a mount message.

This statement can be used only if the user is certain that a removable volume, rather than a fixed volume, will be allocated by the scheduler. If there is any chance that a fixed volume will be allocated, this statement must not be used.

DD EXAMPLE 3

This DD statement makes a specific request for a private, sharable volume to be mounted on a device.

```
//DD1 DD UNIT=disk,VOLUME=(PRIVATE,SER=(121212)),DISP=OLD
```

This DD statement does not ensure that volume integrity is maintained. It should be used with extreme caution in a multiprogramming environment because there is the possibility that a job step running concurrently might make a specific request for the volume, use the volume, and demount it.

DD EXAMPLE 4

This DD statement makes a specific request for a public, nonsharable volume to be mounted on a device.

```
//DD3 DD UNIT=(disk,,DEFER),VOLUME=SER=789012,DISP=OLD
```

If the volume is already mounted, it is used. The volume remains mounted at the end of the job step, and is not demounted until another job step requires the device on which the volume is mounted.

This DD statement ensures that volume integrity is maintained between jobs; two or more such statements in a single job can allocate the same device.

DD EXAMPLE 5

This DD statement makes a specific request for a public, sharable volume to be mounted on a device.

```
//DD1  DD  UNIT=disk,VOLUME=SER=654321,DISP=OLD
```

If the volume is already mounted, it is used. The volume remains mounted at the end of the job step, and is not demounted until another job step requires the device on which the volume is mounted. (This DD statement can also be used to define permanently resident devices.)

This DD statement does not ensure that the volume integrity of a mountable volume is maintained. It should be used with extreme caution in a multiprogramming environment because there is the possibility that a job step running concurrently might use the device.

APPENDIX C. PROCESSING USER LABELS

User labels can be processed by IEBCOMPR, IEBGENER, IEBPTPCH, IEBTCRIN, IEBUPDTE, and IEHMOVE. In some cases, user-label processing is automatically performed; in other cases, you must indicate the processing to be performed. In general, user label support allows the utility program user to:

- Process user labels as data set descriptors.
- Process user labels as data.
- Total the processed records prior to each WRITE command (IEBGENER and IEBUPDTE only).

For either of the first two options, the user must specify standard labels (SUL) on the DD statement that defines each data set for which user-label processing is desired. For totaling routines, OPTCD=T must be specified on the DD statement.

The user cannot update labels by means of the IEBUPDTE program. This function must be performed by a user's label processing routines. IEBUPDTE will, however, allow you to create labels on the output data set from data supplied in the input stream. See "LABEL Statement" on page 303 of the chapter "IEBUPDTE Program."

IEHMOVE does not allow exits to user routines and does not recognize options concerning the processing of user labels as data. IEHMOVE always moves or copies user labels directly to a new data set. See "IEHMOVE Program" on page 361.

Volume switch labels of a multivolume data set cannot be processed by IEHMOVE, IEBGENER, or IEBUPDTE. Volume switch labels are therefore lost when these utilities create output data sets. To ensure that volume switch labels are retained, process multivolume data sets one volume at a time.

PROCESSING USER LABELS AS DATA SET DESCRIPTORS

When user labels are to be processed as data set descriptors, one of the user's label processing routines receives control for each user label of the specified type. The user's routine can include, exclude, or modify the user label. Processing of user labels as data set descriptors is indicated on an EXITS statement with keyword parameters that name the label processing routine to be used.

The EXIT keyword parameters indicate that a user routine should receive control each time the OPEN, EOV, or CLOSE routine encounters a user label of the type specified.

Figure 154 on page 447 illustrates the action of the system at OPEN, EOV, or CLOSE time. When OPEN, EOV, or CLOSE recognizes a user label and when SUL has been specified on the DD statement for the data set, control is passed to the utility program. Then, if an exit has been specified for this type of label, the utility program passes control to the user routine. The user's routine processes the label and returns control, along with a return code, to the utility program. The utility program then returns control to OPEN, EOV, or CLOSE.

This cycle is repeated up to eight times, depending upon the number of user labels in the group and the return codes supplied by the user's routine.

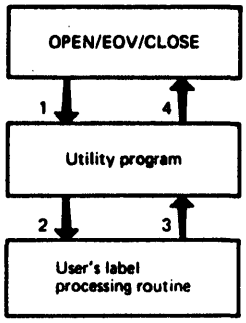


Figure 154. System Action at OPEN, EOV, or CLOSE Time

EXITING TO A USER'S TOTALING ROUTINE

When an exit is taken to a user's totaling routine, an output record is passed to the user's routine just before the record is written. The first halfword of the totaling area pointed to by the parameter contains the length of the totaling area, and should not be used by the user's routine. If the user has specified user label exits, this totaling area (or an image of this area) is pointed to by the parameter list passed to the appropriate user label routine.

An output record is defined as a physical record (block), except when IEBGENER is used to process and reformat a data set that contains spanned records.

The code returned by the user's totaling routine determines system response as follows.

Codes	Meaning
00 (00 hex)	Processing is to continue, but no further exits are to be taken.
04 (04)	Normal processing is to continue.
08 (08)	Processing is to terminate, except for EOD processing on the output data set (user label processing).
16 (10)	Processing is to be terminated.

Figure 155. User Totaling Routine Return Codes

PROCESSING USER LABELS AS DATA

When user labels are processed as data, the group of user labels, as well as the data set, is subject to the normal processing done by the utility program. The user can have labels printed or punched by IEBPTPCH, compared by IEBCOMPR, or copied by IEBGENER.

To specify that user labels are to be processed as data, include a LABELS statement in the job step that is to process user labels as data.

There is no direct relationship between the LABELS statement and the EXITS statement. Either or both can appear in the control statement stream for an execution of a utility program. If there are user label-processing routines, however, their return codes may influence the processing of the labels as data. In addition, a user's output label-processing routine can override the action of a LABELS statement because it receives control before each output label is written. At this time the label created by the utility as a result of the LABELS statement is in the label buffer, and the user's routine can modify it.

A

B

C

D

E

F

G

H

I

J

K

L

M

T

U

V

W

2

3

GC26-4065-1

MVS/370 Utilities
GC26-4065-1

Reader's Comment Form

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Your comments will be sent to the author's department for whatever review and action, if any, are deemed appropriate.

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

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

List TNLs here:

If you have applied any technical newsletters (TNLs) to this book, please list them here:

Last TNL _____

Previous TNL _____

Previous TNL _____

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

GC26-4065-1

Reader's Comment Form

Fold and tape | Please do not staple | Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

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

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
P.O. Box 50020
Programming Publishing
San Jose, California 95150

Fold and tape | Please do not staple | Fold and tape

IBM®

GC26-4065-1

Technical Newsletter

This Newsletter No. GN26-8133
Date 30 March 1984

Base Publication No. GC26-4065-1
File No. S370-32

Prerequisite Newsletters None

MVS/370 Utilities

This technical newsletter (TNL) provides replacement pages for the subject publication to support Release 1.1 of MVS/370 Data Facility Product, Program Product 5665-295. These replacement pages remain in effect for any subsequent releases unless specifically altered. Pages to be inserted and/or removed are:

cover-2
9,10
95-96.1 (96.1 added)
165-172.2 (172.1 to 172.2 added)
183-208.2 (208.1 to 208.2 added)
213-214.1 (214.1 added)

Each technical change is indicated by a vertical bar to the left of the change.

Summary of Amendments

Technical changes effective with this newsletter are noted in the "Summary of Amendments" following the preface.

Note: Please file this cover letter at the back of the publication to provide a record of changes.

IBM Corporation, P.O. Box 50020, Programming Publishing, San Jose, California 95150

Printed in U.S.A.