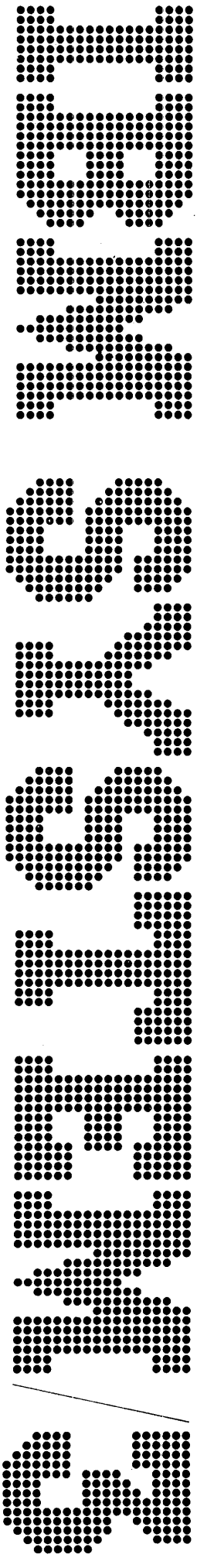


IBM System/3 Model 12 System Control Programming Reference Manual

Program Number 5705-SC1

GC21-5130-0
File No. S3-36

First Edition (March 1976)

This edition applies to version 01, modification level 00 of the IBM System/3 Model 12 System Control Program and to all subsequent versions and modifications until otherwise indicated in new editions or technical newsletters. Changes are periodically made to the information herein; before using this publication in connection with the operation of IBM systems, refer to the latest *IBM System/3 Bibliography,* GC20-8080, for the editions that are applicable and current.

Requests for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A Reader's Comment Form is at the back of this publication. If the form has been removed, address your comments to IBM Corporation, Publications, Department 245, Rochester, Minnesota 55901.

This manual provides the programmer with the information he needs to run programs and to use system utility programs for doing such jobs as preparing disks for use or updating system libraries.

This publication contains two parts. Part 1 describes operation control language (OCL) statements; Part 2 describes system utility programs. For information on the System/3 character sets, see the appendixes.

SYSTEM/3 MODEL 12

System/3 Model 12 is supported by system control programming (SCP) and program products (PPs). The system control programs and program libraries are resident on the attached 3340 Direct Access Storage Facility.

Two program levels are supported if the dual program feature (DPF) is present. The scheduling and controlling of programs in the levels is controlled by operation control language (OCL) statements.

Model 12 provides a print spool function that enables the user to group related print jobs on the print queue. Spooling provides greater flexibility in job scheduling and removes many I/O device conflicts between program levels.

Support for the directly attached 3741 Data Station/Programmable Work Station is similar to that for a card reader or card punch. In this manual, unless otherwise noted, references to card I/O also apply to the directly attached 3741.

RELATED PUBLICATIONS

- *IBM System/3 Model 12 Introduction,* GC21-5116
- *IBM System/3 Model 12 Operator's Guide,* GC21-5144
- *IBM System/3 Model 12 User's Guide,* GC21-5142
- *IBM System/3 Model 12 Halt Guide,* GC21-5145
- *IBM System/3 RPG II Reference Manual,* SC21-7504
- *IBM System/3 RPG II Additional Topics Programmer's Guide,* GC21-7567
- *IBM System/3 Subset American National Standard COBOL Reference Manual,* GC28-6452
- *IBM System/3 Subset American National Standard COBOL Compiler and Library Programmer's Guide,* SC28-6459
- *IBM System/3 Disk FORTRAN IV Reference Manual,* SC28-6874
- *IBM System/3 Models 6, 8, 10, and 12 System Generation Reference Manual,* GC21-5126

Contents

Introduction to OCL Statements

Operation control language (OCL) is your means of communication with the IBM System/3 Model 12 System Control Program. You must write a set of OCL statements for each program you want to run. Based on the information supplied by the OCL statements, the System Control Program will load and run your programs or perform system utility functions.

System control programs must be in main storage before your jobs can be run. These programs are located on disk and are brought into storage by a procedure called initial program load (IPL). IPL is performed by the operator when the system is powered on. For more information on IPL, see *IBM System/3 Model 12 Operator's Guide,* GC21-5142.

The DATE statement is part of the IPL process and must precede the first LOAD or CALL statement of your program. (See *DATE Statement* under *Statement Descriptions* for more information.)

ORGANIZATION OF PART 1

Part 1 is divided into:

- Coding Rules. Defines the general contents of the OCL statements and explains the rules for writing the statements.
- Statement Descriptions. Explains the functions, format, and contents of each OCL statement, and the places in the job stream where the statement may be used.
- Statement Examples. Presents and explains a job stream containing most of the OCL statements.

Coding Rules

TYPES OF INFORMATION

Operation control language (OCL) statements contain, at most, three types of information: a name or comment, a statement identifier, and parameters. A name on the LOAD or JOB statement supplies a label to the unit of work (a job or a job group). The comment allows you to assign a statement identifier for ready reference. A statement identifier distinguishes one statement from another. A parameter is additional information supplied with the statement identifier. Figure 1 shows the general form of OCL statements.

```
// Name or Comment   Identifier   Parameter 1, Parameter 2, ..., Parameter n
```

Figure 1. General Form of OCL Statements

Name

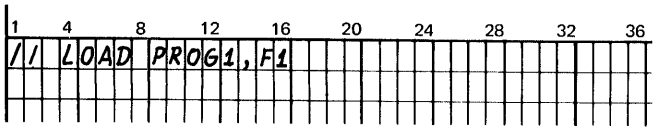
The name is required only on the JOB statement. It is also used by the system if given on a LOAD statement.

Statement Identifiers

Every OCL statement needs one of the following identifiers:

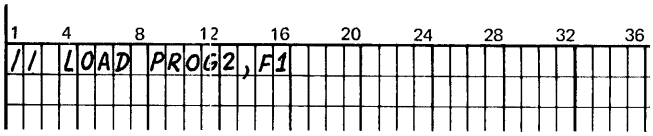
BSCA	JOB	PUNCH
CALL	LOAD	READER
COMPILE	LOCKOUT	RUN
DATE	LOG	SIMULATE
FILE	NOHALT	SWITCH
FORMS	PARTITION	/&
HALT	PAUSE	* (asterisk)
IMAGE	PRINTER	

LOAD is an example of a statement identifier.



Parameters

Some statements need parameters; others do not. (See *Statement Descriptions* for an explanation of the statements that need parameters.) Parameters can be either *codes* or *data.* A code is a word or group of characters that has a certain meaning. Data is information such as the names, locations, and lengths of files on disk. (See *Statement Descriptions* for data and code restrictions on parameters.) In the following example, PROG2 is the name of an RPG II object program, and F1 is a code that stands for simulation area F1 on drive 1, PROG2 is a data parameter and F1 is a code parameter. (For additional information on simulation, see *Simulation on 3340* in the *IBM System/3 Model 12 User's Guide,* GC21-5142.)



Some statements require certain words in parameters to tell one parameter from another. The words are called *keywords.* Parameters containing keywords are called *keyword parameters.* In Figure 2, NAME-MASTER, PACK-VOL1, and UNIT-R1 are keyword parameters. NAME, PACK, and UNIT are keywords. MASTER and VOL1 are data parameters. R1 is a code parameter. There should always be a hyphen between the keyword and the code or data parameter.

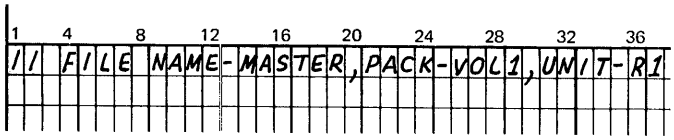


Figure 2. Keyword Parameters

GENERAL CODING RULES

In Part 1 of this manual, the numbers that appear above statement formats and examples indicate the card columns or line positions occupied by the statements. In statement formats, special characters, such as //, and words written in capital letters are information that must be used exactly as shown. Words written in small letters, such as code, program-name, and unit, represent information that you must supply.

Braces ({ }) sometimes appear in parameters shown in statement summaries and parameter summaries. They are not part of the parameters. They simply indicate that you must choose one of several values to complete the parameter. For example, RETAIN-{T/P} means you can use either RETAIN-T or RETAIN-P.

Statements Beginning with //

The rules for coding the statements are as follows (the term *position* refers to either card column or line position):

- Place the // in positions 1 and 2.
- Leave one or more blanks between the // and the word that forms the statement identifier (LOAD, RUN, CALL, etc).
- Leave one or more blanks between the end of the statement identifier and the first parameter.
- If you need more than one parameter, use a comma to separate them. No blanks are allowed within or between parameters. (For the exception to this rule, see the description for the HIKEY parameter under *FILE Statement (Disk)).* Anything following the first blank is considered a comment (see *Comments*).
- If you are writing keyword parameters, place the keyword first and use a hyphen to separate the keyword from the code or data parameter.
- If the parameter is not a keyword parameter, write the parameters in the order in which they are discussed in this manual.

Figure 3 illustrates the coding rules. The statement identifiers are LOAD and FILE. The parameters are PROG1, R1, NAME-MASTER, UNIT-R1, and PACK-VOL1. The last three parameters are keyword parameters.

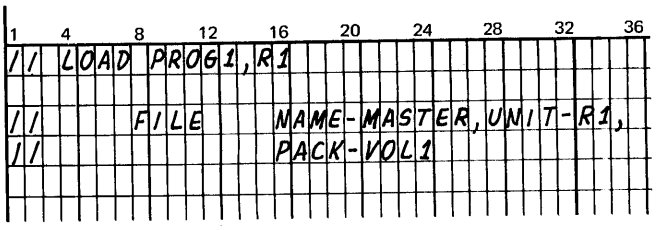


Figure 3. Illustration of General Coding Rules

Statements not Beginning with //

* and /& statements do not require // preceding them when coded. (See *Statement Descriptions* for * and /& statements.)

Continuation

All OCL statements except FILE, PRINTER, COMPILE, and FORMS must not exceed 96 characters, including blanks and comments. (Data for the IMAGE statement requires continuation for the record containing the chain image characters, but the data follows different continuation rules. See *IMAGE Statement* under *Statement Descriptions* for more information.)

The continuation rules are as follows:

- Place a comma after the last parameter in every record except the last. The comma, followed by a blank, tells the system that the statement is continued in the next record.
- Begin each new record with a // in positions 1 and 2.
- Leave one or more blanks between the // and the first parameter in the record. (See *HIKEY Parameter* under *FILE Statement (Disk)* for exception to this rule.)

Figure 4 illustrates the continuation rules.

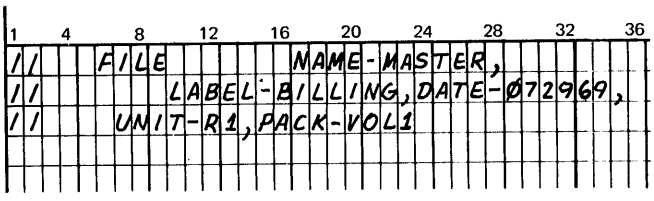


Figure 4. Illustration of Continuation Rules

Comments

You can include comments in the following places in your statements:

- Following the // in statements beginning with //. Begin the comment in position 3, immediately following the //. You can use up to 8 characters without blanks. Leave one or more blanks between the comment and the word forming the statement identifier. Figure 5 contains such a comment. The word BILLING is the comment. (On a JOB statement the word BILLING is the groupname, and on a LOAD statement BILLING is the jobname.)

- After the last parameter. Leave one or more blanks between the last parameter and your comment. The comment can be any combination of characters. If the statement is continued in subsequent cards or lines, you can place comments after the last parameter in any of the cards or lines.

- After statements without parameters. Leave one or more blanks between the statement identifier and your comment. Examples of statements without parameters are: /&, // PAUSE, and // RUN. (Statements with all parameters optional and unused cannot contain this type of comment.)

In addition to writing comments within your OCL statements, you can include whole cards or lines of comments. The OCL comment statement is provided for that purpose. (See * *(Comment) Statements* under *Statement Descriptions* for more information.)

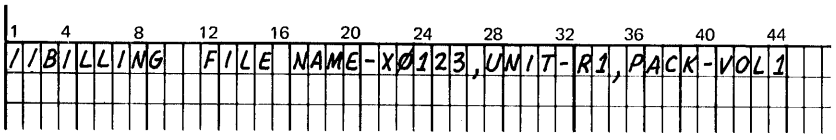


Figure 5. Comment Following //

Statement Descriptions

Each OCL statement is described separately in this section. The following information is given for each statement:

- The function of the statement.

- The placement of the statement in regard to other statements and the circumstances under which the statement is needed.

- The format of the statement.

- The contents of the statement, explaining the parameters that can be used in the statement.

Figure 6 gives the function, placement, and restrictions on use for each OCL statement.

Figure 7 describes the contents of the OCL statements. It is meant for reference only. If you are not familiar with an entry, or you do not know when to use or omit it, refer to the proper statement in the remainder of this section.

When using Figure 7, remember that words written in small letters (such as filename or value) require a choice on your part, depending on the functions you want the statement to perform. Refer to Figure 7 to see which parameters are available. Capitalized parameters must be coded along with the data or code parameter.

Statement	Function	Placement		Restrictions On Use
		Statement Appears in Job Stream	Statement Appears in a Procedure	
// BSCA	Changes the BSCA line number.	Must follow LOAD or CALL statement and precede the RUN statement.	Must follow the LOAD statement and precede the RUN statement (if RUN is used).	
// CALL	Identifies procedure to be merged into job stream and the simulation area containing the source library from which to read the procedure.	Must precede the RUN statement.	Indicates chained procedures.	Can be no more than nine levels of nested chained procedures.
// COMPILE	Tells the system where the source program to be compiled is located and where to place the object program.	Must follow LOAD or CALL statement and precede the RUN statement.	Must follow the LOAD statement and precede the RUN statement (if RUN is used).	
// DATE	Supplies the system with a date; this date is given to disk files being created.	Must follow LOAD or CALL statement and precede the RUN statement except at IPL time, when it must precede the first LOAD or CALL statement.	Must follow the LOAD statement and precede the RUN statement (if RUN is used).	Must be supplied during the initial program load. If used after IPL, the effect of the statement is for that job only.
// FILE	Supplies information about the file to the system.	Must follow LOAD or CALL statement and precede the RUN statement.	Must follow the LOAD statement and precede the RUN statement (if RUN is used).	Required for every new file created and existing files being used.
// FORMS	Same as the PRINTER statement.			Cannot be used to override the PRINTER statement in a procedure.
// HALT	Instructs system to halt when program ends; cancels the effect of the NOHALT statement.	Anywhere among the OCL statements.	Must precede the RUN statement (if RUN is used).	
// IMAGE	Tells the system to replace the chain-image area with characters indicated in the following data statements or characters keyed in or read from source library.	Anywhere among the OCL statements.	Must precede the RUN statement (if RUN is used).	Required if the printer chain has been changed.

// JOB
or
//groupname JOB
	Allows you to group print jobs together on the spool file.	Must precede the first LOAD or CALL statement for a group.	Cannot be used in a procedure.	

Figure 6 (Part 1 of 3). Table of OCL Statements

Statement	Function	Placement		Restrictions On Use
		Statement Appears in Job Stream	Statement Appears in a Procedure	

// LOAD
or
//jobname LOAD
	Identifies the program to be run and indicates the simulation area that contains the object library from which it is to be loaded.	Must precede the RUN statement. Must follow the JOB statement (if JOB is used).	Must precede the RUN statement (if RUN is used). Only one LOAD statement is allowed in a procedure.	

// LOAD *
or
//jobname
LOAD *
	Indicates that the object program will be loaded from the system input device following the RUN statement.	Must precede the RUN statement. Must follow the JOB statement (if JOB is used).	Must precede the RUN statement (if RUN is used). Only one LOAD * statement is allowed in a procedure.	LOAD * cannot be used in program level 2.

// LOCKOUT	Disables the other program level to allow fast job initiation in the program level in which the LOCKOUT statement was read.	Anywhere among the OCL statements.	Must precede the RUN statement (if RUN is used).	Ignored on a non-DPF system.
// LOG	Instructs system to start or stop printing OCL statements and codes, indicates the device to be used to print them, and controls page eject at the end of job.	Anywhere among the OCL statements.	Must precede the RUN statement (if RUN is used).	Device cannot be specified in program level 2.
// NOHALT	Instructs system to continue without stopping when a program ends. Causes certain halts to default.	Anywhere among the OCL statements	Must precede the RUN statement (if RUN is used).	
// PARTITION	Guarantees a minimum size to level 2 for a program in that level.	Anywhere among the OCL statements.	Must precede the RUN statement (if RUN is used).	Cannot be submitted in program level 2 or when program level 2 is processing.
// PAUSE	Tells the program to stop in order to give the operator time to perform a function. Operator must restart program.	Anywhere among the OCL statements.	Must precede the RUN statement (if RUN is used).	
// PRINTER	Enables you to describe the functions performed by the system print device and control options related to print spooling.	Anywhere among the OCL statements.	Must precede the RUN statement (if RUN is used). Cannot be used to override the FORMS statement in a procedure.	
// PUNCH	Enables you to change the system punch device.	Anywhere among the OCL statements.	Must precede the RUN statement (if RUN is used).	

Figure 6 (Part 2 of 3). Table of OCL Statements

Statement	Function	Placement		Restrictions On Use
		Statement Appears in Job Stream	Statement Appears in a Procedure	
// READER	Changes the system input device used to read OCL statements.	Must precede LOAD or CALL statement or follow the RUN statement and precede the next LOAD or CALL statement.	Must precede the LOAD statement (if LOAD is used).	In a procedure, OCL statements are not read from the input device until the procedure is completely executed.
// RUN	Indicates the end of the OCL statements for a program and tells the system to run the program.	Must be the last OCL statement.	May be the last statement.	Required in the job stream for each program which is to be run.
// SIMULATE	Instructs the system to turn simulation ON or OFF on D2. This also enables or disables R2 and F2.	Must not come between a LOAD or CALL and a RUN.	Cannot be used in a procedure.	Other program level must be at end of job. A rollin cannot be pending.
// SWITCH	Used to set one or more external indicators on or off or leave the indicator as it is.	Must follow LOAD or CALL statement and precede the RUN statement.	Must follow the LOAD statement and precede the RUN statement (if RUN is used).	
/&	Provides OCL security from previous job.	Recommended as the first statement of a job.	Not allowed in a procedure.	Can be used in the job stream only.
* (Comment)	Used to explain the job or give the operator instructions; does not affect the program in operation.	Anywhere.	Anywhere.	

Figure 6 (Part 3 of 3). Table of OCL Statements

Statement	Parameter	Code	Meaning of Code
// BSCA	LINE	LINE-1 2	Change all BSCA DTF line codes to the line number specified.
// CALL	procedure name	name	Name that identifies the procedure in the source library.
	unit	R1 R2 F1 F2	Simulation area containing the procedure (see note).
// COMPILE	SOURCE	SOURCE-name	Name of source program.
	UNIT	UNIT-R1 R2 F1 F2	Simulation area that contains the source library (see note).
	OBJECT	OBJECT-R1 R2 F1 F2	Where to place the object program (see note). Does not apply to object program placement for COBOL or FORTRAN compilers.
// DATE	date	mmddyy or ddmmyy	System date or date within a set of statements.
// FILE (Disk Files)	NAME	NAME-filename	Name the program uses to refer to the file.
	UNIT	UNIT-R1 R2 F1 F2	Simulation area that contains or will contain the file (see note).
		D1 D2	Location of the main data area that contains or will contain the file.
	PACK	PACK-name	Name of disk that contains or will contain the file.
	LABEL	LABEL-filename	Name by which your file is identified on disk.
	RECORDS or TRACKS	RECORDS-number of TRACKS-number	Amount of space needed on a disk for a file.
	LOCATION	LOCATION-track number	Number of track on which file begins or is to begin (simulation area only).
		LOCATION-cylinder number	Cylinder number on which file begins or is to begin. Track assumed zero (main data area only).
		LOCATION-cylinder number/track number	Cylinder number, track number on which file begins or is to begin (main data area only).
	RETAIN	RETAIN-T S P A	Temporary file Scratch file Permanent file Reactivate scratch file (simulation area only).

Note: For an explanation of the unit codes, see *Simulation Area* in the *IBM System/3 Model 12 User's Guide,* GC21-5142.

Figure 7 (Part 1 of 5). Table of Parameters

Statement	Parameter	Code	Meaning of Code
	DATE	DATE-mmddyy ddmmyy	The date the file was created.
	HIKEY	HIKEY-'highest key fields allowed'	List of highest key fields allowed on each pack.
// FILE (Tape File)	NAME	NAME-filename	Name that the program uses to refer to the file.
	UNIT	UNIT-T1 T2 T3 T4	Where the tape that contains or will contain the file is mounted.
	REEL	REEL-name	Name of the tape that contains or will contain the file.
		-NL	The tape is not labeled.
		-NS	The tape contains non-standard labels.
		-BLP	Bypass label processing of standard labeled tapes.
	LABEL	LABEL-filename or LABEL-'character string'	Name by which your file is identified on tape.
	DATE	DATE-mmddyy ddmmyy	Tells the system the date the file was created.
	RETAIN	RETAIN-nnn	The number of days a file should be retained before it expires.
	BLKL	BLKL-block length	The number of bytes in a physical block of tape.
	RECL	RECL-record length	The number of bytes in a logical record.
	RECFM	RECFM-F	Fixed length, unblocked records.
		-V	Variable length, unblocked records.
		-D	Variable length, unblocked, D-type ASCII records.
		-FB	Fixed length, blocked records.
		-VB	Variable length, blocked records.
		-DB	Variable length, blocked, D-type ASCII records.
	END	END-LEAVE	The tape remains in its present position after the file is processed.
		-UNLOAD	The tape is rewound and unloaded after processing.
		-REWIND	The tape is rewound after processing.

Figure 7 (Part 2 of 5). Table of Parameters

Statement	Parameter	Code	Meaning of Code
	DENSITY	DENSITY-200	The tape will be written at 200 bpi (bits per inch) density.
		-556	The tape will be written at 556 bpi density.
		-800	The tape will be written at 800 bpi density.
		-1600	The tape will be written at 1,600 bpi density.
	ASCII	ASCII-YES	An ASCII file is being processed.
		-NO	An EBCDIC file is being processed.
	DEFER	DEFER-YES	The tape volume will be mounted later.
		-NO	The tape is presently mounted.
	CONVERT	CONVERT-ON	Data read from or written to a 7-track tape file will be converted.
		-OFF	Data read from or written to a 7-track tape file will not be converted.
	TRANSLATE	TRANSLATE-ON	Data read from or written to a 7-track tape file will be translated.
		-OFF	Data read from or written to a 7-track tape file will not be translated.
	PARITY	PARITY-EVEN	The 7-track tape file will be read or written in even parity.
		-ODD	The 7-track tape file will be read or written in odd parity.
// FORMS	Same as PRINTER statement.	Same as PRINTER statement.	Same as PRINTER statement.
// HALT	none		
// IMAGE	format	HEX	Indicates characters from the system input device are in hexadecimal form.
		CHAR	Indicates characters from the system input device are in EBCDIC form.
		MEM	Indicates characters are from the source library.
	number	value	Number of new characters.
	name	name	Identifies the characters in the library.
	unit	R1 R2 F1 F2	Simulation area that contains the library (see note).

Figure 7 (Part 3 of 5). Table of Parameters

Statement	Parameter	Code	Meaning of Code
//groupname JOB	PRIORITY	PRIORITY-{0, 1, 2, 3, 4, 5}	Specifies the priority of jobs in the spool file. Default is 1.
	SPOOL	SPOOL-{YES, NO}	Indicates whether the group of jobs identified by the groupname is to be spooled. Default is YES.

// LOAD
or
//jobname LOAD
	asterisk	*	Program is to be loaded from the system input device.

// LOAD
or
//jobname LOAD
	program name	name	Name of program that is to be loaded from disk.

	unit	R1 R2 F1 F2	Simulation area that contains the object library (see note).
// LOCKOUT	none		
// LOG	code	CONSOLE	Use printer-keyboard as logging device.
		PRINTER	Use printer as logging device.
		OFF	Stop printing.
		ON	Start printing.
	mode	EJECT	Eject a page at end of job.
		NOEJECT	Suppress page eject at end of job.
			When you use the spool writer, an eject occurs at the start of every job, regardless of the mode specified in the LOG statement.
// NOHALT	SEVERITY	SEVERITY-{1, 2, 4, 8}	Tells the system to select default options for error halts.
// PARTITION	size	value	Minimum size of program level 2 in decimal bytes.
// PAUSE	none		
// PRINTER	DEVICE	DEVICE-{5203, 5203L, 5203R}	5203/5203L specifies left carriage 5203 or 1403. 5203R specifies right carriage 5203.
	LINES	LINES-number	Specifies the number of print lines per page.
	FORMSNO	FORMSNO-forms number	Informs the operator which forms type should be mounted on the printer.

Note: For an explanation of the unit codes, see *Simulation Area* in the *IBM System/3 Model 12 User's Guide,* GC21-5142.

Figure 7 (Part 4 of 5). Table of Parameters

Statement	Parameter	Code	Meaning of Code
	COPIES	COPIES-number	With spooling active, allows you to obtain more than one copy of each job's printed output.
	DEFER	DEFER-YES/NO	Allows you to begin printing a job's spooled output before the job completes execution (DEFER-NO). Default is DEFER-YES.
	ALIGN	ALIGN-YES/NO	Allows you to perform forms alignment for spooled printed output (ALIGN-YES). Default is ALIGN-NO.
// PUNCH	system punch device	MFCU2	Secondary hopper of MFCU.
		MFCU1	Primary hopper of MFCU.
		1442	Card Read/Punch.
		3741	Data Station/Programmable Work Station.
// READER	system input device	CONSOLE	Printer-keyboard.
		MFCU2	Secondary hopper of MFCU.
		MFCU1	Primary hopper of MFCU.
		1442	Card Read/Punch
		3741	Data Station/Programmable Work Station.
// RUN	none		
// SIMULATE	status	ON OFF	Enables/disables simulation on drive 2. // SIMULATE OFF allows offline multivolume files to be processed on D2.
// SWITCH	indicator-settings	Refer to *SWITCH Statement* under *Statement Descriptions*	
/&	none		
* (Comment)	none		

Figure 7 (Part 5 of 5). Table of Parameters

BSCA STATEMENT

Function

The BSCA statement allows you to change all BSCA (binary synchronous communications adapter) line specifications in your program; therefore, you can use either BSCA line without recompiling the program. (The program must have been compiled on a system that had both BSCA lines specified during system generation.) If the BSCA statement is not entered, the line specifications in the program are not changed.

Placement

The BSCA statement must follow the LOAD or CALL statement and precede the RUN statement.

Format

// BSCA parameter.

Content

The parameter is a keyword parameter. The parameter is LINE-code. The codes are as follows:

Code	Meaning
1	Change all BSCA line specifications to BSCA line 1.
2	Change all BSCA line specifications to BSCA line 2.

CALL STATEMENT

Function

CALL statements are needed only when you want to merge procedures into the job stream.

To understand the function of the CALL statement, you must understand the relationship between the job stream and procedures. The job stream contains the OCL statements that control the system. The system reads them from the system input device. Procedures are sets of OCL statements in a source library on a simulation area. They have no effect on the system until they are merged into the job stream.

You can modify the procedure identified by a CALL statement, by providing other OCL statements (procedure override statements, see *Changing Parameters* in the *IBM System/3 Model 12 User's Guide,* GC21-5142) after the CALL statement. These statements temporarily modify the procedure. The last statement of the CALL sequence must be a RUN statement. The RUN statement is required, however, whether or not you supply other OCL statements. (For further explanations, see *Procedures* in the *IBM System/3 Model 12 User's Guide,* GC21-5142.)

Placement

CALL statements can be used in the job stream or in a procedure. They are, in effect, replaced by the procedures they identify. The last statement of the CALL sequence must be a RUN statement.

Format

```
// CALL procedure-name,unit
```

Content

Procedure-name: The procedure-name is the name that identifies the procedure in the source library. You supply the procedure-name in the Library Maintenance control statements when you use the program to place the procedure in the library. (See *Library Maintenance* in Part 2 of this manual for restrictions on procedure-name.)

Unit: The unit parameter is a code indicating which simulation area contains the procedure. The codes are R1, F1, R2, and F2.

COMPILE STATEMENT

Function

The COMPILE statement tells the system two things: (1) where the source program to be compiled is located if it is coming from a source library; (2) where the object program is to be placed. (An object program is a source program that has been compiled or translated into machine language.)

Placement

The COMPILE statement must be within the set of OCL statements that apply to the compilation. The COMPILE statement must follow the LOAD or CALL statement and precede the RUN statement.

Format

```
// COMPILE parameters
```

Content

All the parameters are keyword parameters (keywords are in capital letters). The keywords are: SOURCE, UNIT, and OBJECT.

SOURCE: The SOURCE parameter tells the system the name of the source program. The keyword SOURCE must be followed by the name of a source program. The name is the name by which the source program is identified in the source library.

You can place source programs in a source library by using the Library Maintenance program. The program name you supply in Library Maintenance control statements is the name used to identify the source program in the library. (For more information, see *Library Maintenance* in Part 2 of this manual.)

If the SOURCE parameter is not used, the source program is assumed to be in the job stream following the RUN statement.

The SOURCE parameter must always be accompanied by the UNIT parameter.

UNIT: The UNIT parameter is used only when the SOURCE parameter is used.

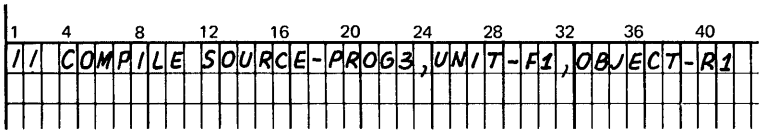
The UNIT parameter is a code indicating which simulation area contains the source library. The codes are R1, F1, R2, and F2.

OBJECT: The OBJECT parameter tells the system where to place the object program. The OBJECT parameter may be specified without using the SOURCE and UNIT parameters. The codes which are used to indicate the simulation area in which the object program is to be placed are R1, F1, R2, and F2.

Note: If the OBJECT parameter is omitted, it is assumed that the object program is to be placed in the same simulation area as the compiler. The OBJECT parameter does not affect FORTRAN or COBOL object programs.

Example

The following sample COMPILE statement tells the system that the source program with the name PROG3 is located in the simulation area F1 on drive 1.



The parameter OBJECT-R1 tells the system to place the object program in the simulation area R1 on drive 1.

DATE STATEMENT

Function

The DATE statement gives the system a date, called the *system date.* The system date is referred to by RPG II field names UDATE, UMONTH, UDAY, and UYEAR. The preceding field names can also be used in a reference to the date given to the disk files when they were created.

A DATE statement within the set of statements for a program changes the system date, but only for that program. When the program ends, the date supplied in the DATE statement at IPL time is again used. There can only be one DATE statement per job.

Placement

A DATE statement is always required before the first LOAD or CALL statement after initial program load (IPL).

A DATE statement can also appear within any of the sets of statements for your programs. The DATE statement must follow the LOAD or CALL statement and precede the RUN statement.

Format

// DATE date

Content

The system date can be in either of two formats: month-day-year (mmddyy) or day-month-year (ddmmyy). You must specify the format at system generation time. (See *IBM System/3 Models 6, 8, 10, and 12 System Generation Reference Manual,* GC21-5126, for more information on system generation.) The date you specify must be in that format.

Example

The date can be written with or without punctuation. For example, February 25, 1976, could be specified in any one of the following ways:

02-25-76
25-02-76
022576
250276

Month, day, and year must each be two-digit numbers, but leading zeros in month and day may be omitted when punctuation is used (2-25-76 or 25-2-76). In the punctuated format, any characters except commas, quotes, numbers, and blanks can be used as punctuation.

FILE STATEMENT (DISK)

Function

The FILE statement provides information about the files on a data module so that disk system management can read and write records for user programs.

Placement

The 3340 is referenced through OCL statements at execution time. During operation in a DPF environment on the Model 12, either or both drives can be addressed by both program levels, but the same file cannot be addressed by both program levels at the same time unless:

- Both program levels are using a file as input only
- One program level is using a file as input and the other is using it as update

Files can reside in the main data area or in the simulation areas. A FILE statement must be provided for each file used by your programs. It must be between the LOAD and RUN or CALL and RUN statements for each program using the 3340. Split cylinder files are not supported on the 3340. The maximum number of files allowed is explained in *Maximum Number of Files* in the *IBM System/3 Model 12 User's Guide,* GC21-5142.

Format

// FILE parameters

Content

Figure 8 summarizes the keywords of the FILE statement. The following sections provide additional information about the keyword parameters.

Keyword	Parameter	Keyword Required or Optional
NAME	Filename	Required
PACK	Name	Required
UNIT	Code	Required
LABEL	Filename	Optional
DATE	Date	Optional
RETAIN	Code	Optional
RECORDS or TRACKS	Number Number	Required for creating files
LOCATION	Cylinder number (main data area only)	Optional
	Cylinder number/ track number (main data area only)	Optional
	Track number (simulation area only)	Optional
HIKEY (main data area only)	Highest allowed key fields	Optional

Figure 8. Description of Parameters on the OCL FILE Statement for the 3340

Keyword Parameters for Single Volume Disk Files

NAME: The NAME parameter is required for the FILE statement. It informs disk system management of the name that your program uses to refer to the file. The filename can be any combination of characters except commas, apostrophes, or blanks. The first character must be alphabetic and the number of characters must not exceed 8.

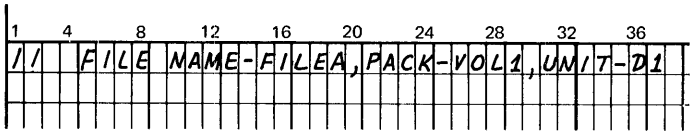
The following list shows the program, the type of file, and the file name:

Program	File	Name
Copy/Dump	Input	COPYIN
	Output	COPYO
Disk Sort	Input	INPUT
	Work	WORK (optional)
	Output	OUTPUT
Assembler	Input	$SOURCE
	Output	$WORK
	Work	$WORK 2
COBOL Compiler	Input	$SOURCE
	Work	$WORK
	Work	$WORKX
FORTRAN Compiler	Input	$SOURCE
	Work	$WORK
RPG II Compiler	Input	$SOURCE
	Work	$WORK
1255 Utility	Output	F1255
RPG II Auto Report	Input	$SOURCE
	Work	$WORK
Macro Processor	Output	$SOURCE
Overlay Linkage Editor	Input	$SOURCE
	Work	$WORK
Any program using large indexed files	Work	$INDEX45 (for main data area file)

These files must be in a simulation area

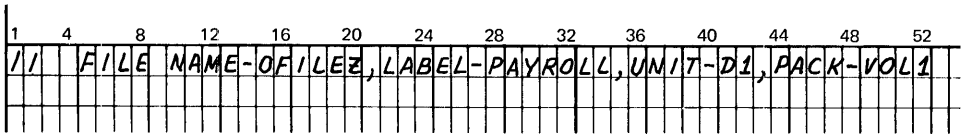
PACK: The PACK parameter is also required for the FILE statement. It informs disk system management of the name of the main data area or simulation area that contains or will contain the file. The management routines check this name to ensure that it is the same as the name in the volume label of the area being used. This parameter can consist of from 1 to 6 characters, excluding the apostrophe, comma, and blank.

UNIT: The UNIT parameter is the last of the required parameters in the FILE statement. It supplies the location of the main data area or simulation area that contains the file. The possible codes are F1, R1, F2, R2, D1, and D2.



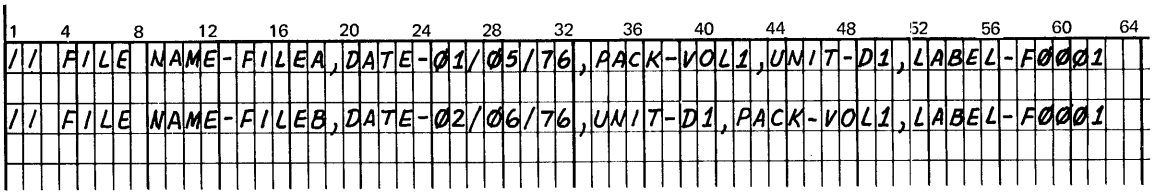
The preceding example shows how the UNIT parameter for a file located in the main data area on drive 1 would be coded.

LABEL: The LABEL parameter refers to the filename by which the file is identified in the VTOC. This parameter is required only if the filename in a program differs from the filename on the main data area or simulation area. If a new file is being created and the LABEL parameter is omitted, the filename from the NAME parameter is used.



The preceding example shows how the LABEL parameter for a file named PAYROLL would be coded.

DATE: The DATE parameter is required when two or more files having the same name exist on a main data area or simulation area and a file with a particular date is desired. The creation date of the desired file is coded in the DATE parameter. If two or more files with the same name exist on a main data area or simulation area and neither the date nor the location is given, the file having the latest creation date is selected. The date must be in the format month-day-year or day-month-year as was specified at system-generation time. The date must be written as a six-digit number with three fields of two digits without punctuation, or three fields of one or two digits with the fields separated by punctuation. Any characters except numbers, apostrophes, commas, or blanks can be used as punctuation.

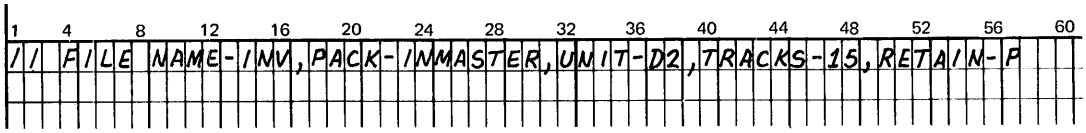


In the preceding example are the NAME, LABEL, and DATE parameters for two versions of a file on the same disk, one written on January 5, 1976, the other on February 6, 1976. Both files have the same label: F0001.

RETAIN: The optional RETAIN parameter indicates the classification of the file when it is created. The classifications are:

Code	Meaning
S	Scratch file. A scratch file is intended for use by the current program and does not exist after the completion of the current program. S is also used to remove a temporary file so that its space will be available to subsequent programs.
T	Temporary file. A temporary file is one that has short-term usefulness and can be overwritten when this usefulness has ended.
P	Permanent file. A permanent file is one that is expected to be maintained permanently on the data module.
A	Reactivate scratch file. RETAIN-A must be coded to reactivate a scratch file, which changes it to a temporary file. This can only be specified for files in the simulation area.

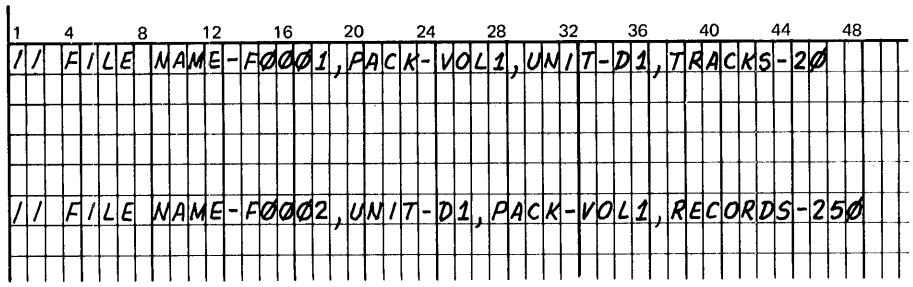
The file is assumed to be temporary if the RETAIN parameter is omitted at file-creation time.



The preceding example shows how the RETAIN parameter is coded for a permanent file.

RECORDS or TRACKS: Either the RECORDS or TRACKS parameter, but not both, can appear in the FILE statement. One of these is required for files being created and indicates the amount of space necessary for the file. If the file is being referenced, these parameters inform disk system management of the amount of space that was used for the file when it was created. The space requirement is specified as the number of records in the file (RECORDS) or as the number of tracks (TRACKS). When more than one file on the same main data area or simulation area has the same filename, this keyword parameter can be used to identify the desired file. Two restrictions are applicable when the space requirement is defined:

- If RECORDS is used, the number can be up to six digits long and must be within the range of 1 through 999999.
- If TRACKS is used, the number can be up to four digits long and must be within the range of 1 through 3320 when the file is in the main data area or 1 through 398 when the file is in a simulation area.



The preceding example shows how the TRACKS parameter for a file requiring 20 tracks is coded and how the RECORDS parameter is coded if a file contains 250 records.

LOCATION: For the main data area, the optional LOCATION parameter is used to specify the cylinder and track on which the file is to start; for a simulation area, this parameter is used to specify the track on which the file is to start. You can specify either the cylinder number or the cylinder number and the track number for the main data area. If the track number is omitted, it is assumed to be zero. For the main data area, the cylinder number must be from 1 through 166 and the track number from 0 through 19. The cylinder number and track number, when specified together, must be separated by a slash (ccc/tt). For a simulation area, the track number must be from 8 through 405.

When you are accessing an existing file, the LOCATION keyword parameter must be identical to that used in creating the file. When you are creating a file, this parameter specifies the beginning position of the file.

When two or more files on the same main data area or simulation area have the same filename, this keyword parameter can be used to identify the desired file.

Keyword Parameters for Multivolume Files

For online multivolume files, the keyword parameters that require lists are PACK, UNIT, TRACKS, RECORDS, LOCATION, and HIKEY. These parameters require lists to describe both data modules containing the file. For offline multivolume files, lists are also used, but UNIT does not require a list since all the volumes must be mounted on the same drive (D2).

You must follow certain rules when indicating the lists for these parameters:

- The lists must be enclosed in quotes.
- The items in the list must be separated by commas.
- The lists, except for HIKEY, must not contain blanks.

The functions of the keyword parameters have been explained (except for HIKEY which is explained here); therefore, only the considerations for using the lists in these parameters are explained here.

PACK: The list for this parameter contains the names of the volumes in the order they are to be used.

UNIT: If the number of units specified for this parameter is less than the number of volumes specified for the PACK parameter, the file is processed as an offline multivolume file.

For online multivolume files, the unit codes must be specified in the sequence of the two volumes used (specified by the PACK parameter).

For offline multivolume files, the unit code is D2. All volumes (specified by the PACK parameter) are processed on D2.

```
PACK-'VOL1,VOL2,VOL3',UNIT-D2
```

Unit D1 cannot be used for offline multivolume files. Unit D2 can be used for offline multivolume files when simulation of R2 and F2 is disabled (via a SIMULATE OCL statement).

TRACKS or RECORDS: The list for these parameters indicates the amount of space occupied by the multivolume file. The numbers in the list must correspond to the order of the names listed in the PACK parameter.

LOCATION: The list for this parameter contains the cylinder number or the cylinder number/track number parameter for the data modules you use for the file. The parameters must correspond to the order of the names in the PACK parameter. If LOCATION is specified for one volume of a multivolume file, it must be specified for all the volumes of that file.

HIKEY: The HIKEY parameter is used only for multivolume indexed files. HIKEY limits the highest key field that can be put on each data module of a multivolume file. For example, in the following HIKEY parameter, three volumes are used: HIKEY-'JONES,NICOL,ZZZZZ'. The highest key field allowed on the first volumes is JONES. This means that all the records up to and including JONES are on this volume. Since HIKEY parameters must be in ascending order, the next volume contains all of the records with keys following JONES and including NICOL. The last volume contains all the records with names that come after NICOL and through ZZZZZ.

OCL considerations for the HIKEY parameter are:

1. All characters except commas are valid.
2. The list of HIKEY parameters must begin and end with an apostrophe even if only one parameter is specified. A single apostrophe in a key field must be written as a double apostrophe in the HIKEY parameter.

3. For each PACK parameter specified, there must be a corresponding HIKEY key field parameter for that pack.

4. The HIKEY fields must be equal in length and must be specified in ascending order.

5. The maximum length of a HIKEY field is 29 characters.

6. The HIKEY fields must be the same length as the keys on file.

7. Continuation of HIKEY sublists must begin in column 4 of the continuation record following the // blank.

8. Comments must not follow the last comma on a FILE statement when the last parameter is an incomplete HIKEY sublist.

Packed HIKEY: The packed HIKEY parameter has all the OCL considerations for HIKEY including the following restrictions:

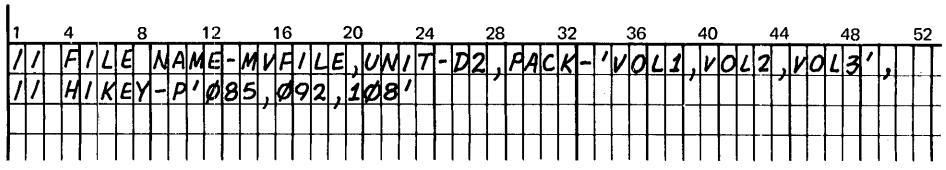
1. The first character following the HIKEY keyword and dash (HIKEY-) must be a P to indicate packed HIKEY.

2. All characters in the packed HIKEY must be zoned numerics (0-9).

3. The number of digits in each packed key must be the same.

4. The number of zoned numeric characters per packed HICKEY must not exceed 15, since the maximum packed key field length is 8.

The following example shows a packed HIKEY parameter. In the example the key field length of MVFILE is 2. The HIKEYs are X'085F', X'092F', and X'108F' for VOL1, VOL2, and VOL3, respectively. The first two packed keys required a leading zero to make the lengths consistent.



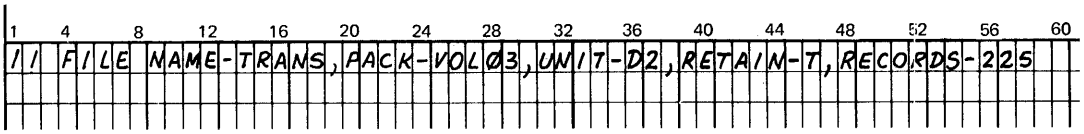
Examples

The following are examples of FILE statements. In each example, the file is described first, then the corresponding FILE statement is shown.

Example 1: Suppose that each week you create a disk file that contains the records for the transactions you had made that week. Assume the following facts about that file:

- The name your program uses to refer to the file is TRANS, which is also the name you want to use to identify the file on disk.
- You are placing the file in a main data area named VOL03.
- You intend to mount the data module on drive 2.
- You want to save the file for use at the end of the month.
- The file contains 225 records.
- You are letting the system choose the area that will contain the file.

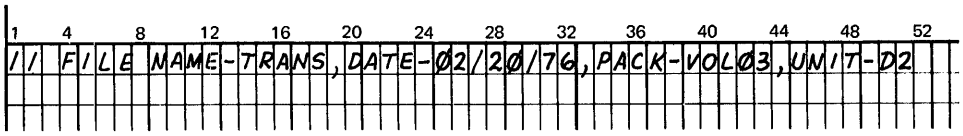
The following example shows how the FILE statement for the preceding file is coded:



Example 2: Suppose you had created, on the same data module (VOL03), four versions of the transaction file described in the preceding example—one for each of the weeks in February, 1976. Assume the following:

- You had created the files on the following days: 2/6/76, 2/13/76, 2/20/76, and 2/27/76 (these were the system dates used for each of the files).
- You want to reference the third file (the one created 2/20/76).
- You intend to mount the data module on drive 2.

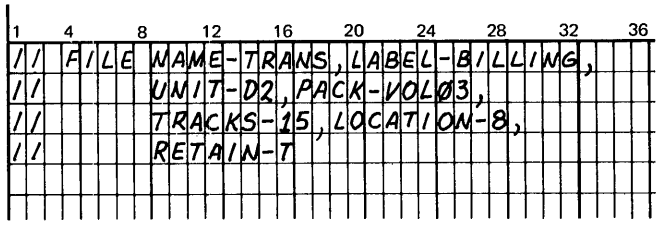
The file statement you would need is:



Example 3: Suppose that at the end of the month you combine the files referred to in example 2, for use in preparing your monthly bills. Further assume the following:

- Your program uses the name TRANS to refer to the file, but you want to use the name BILLING to identify the file on disk.
- You are expressing the amount of disk space as the number of tracks required to contain the file (assume the number is 15), and you want the file to begin on cylinder 8, track 0.
- You are placing the file in a main data area named VOL03.
- You intend to mount the data module on drive 2.

The following example shows the FILE statement you would use for this file.



Example 4: Suppose you want to create two versions of two files on disk and later to access one version of each file. Further assume the following:

- The names your program uses to refer to the files are AA and BB, which are also the names you want to use to identify the files on disk.
- File AA and BB are being placed on a data module on drive 2 named D2D2D2.
- One version of each file is created on 1/12/76 and 1/13/76.

- Disk space and location for the files are:

File	Version	Tracks	Location
AA	1/12/76	10	120/0
	1/13/76	10	130/0
BB	1/12/76	20	140/0
	1/13/76	20	150/0

- You want to access file AA, version 1/12/76, and file BB, version 1/13/76.

The following OCL statements are needed to create the above versions of files AA and BB and to access a version of each file.

```
1   4    8   12   16   20   24   28   32   36   40   44   48
*   CREATES VERSIONS OF FILES AA AND BB

// DATE-01/12/76
// LOAD RPGOBJ,R1
// FILE NAME-AA,UNIT-D2,PACK-D2D2D2,
//      TRACKS-10,LOCATION-120,RETAIN-T
// FILE NAME-BB,UNIT-D2,PACK-D2D2D2,
//      TRACKS-20,LOCATION-140/0
// RUN

*   CREATES ANOTHER VERSION OF FILES AA AND BB

// LOAD RPGOBJ,R1
// DATE-01/13/76
// FILE NAME-AA,UNIT-D2,PACK-D2D2D2,
//      TRACKS-10,LOCATION-130/0
// FILE NAME-BB,UNIT-D2,PACK-D2D2D2,
//      TRACKS-20,LOCATION-150
// RUN

*   ACCESSES FILE VERSIONS OF ABOVE FILES

// LOAD RPGIN,R1
// FILE NAME-AA,UNIT-D2,PACK-D2D2D2,
//      LOCATION-120
// FILE NAME-BB,UNIT-D2,PACK-D2D2D2,
//      DATE-01/13/76
// RUN
```

File Processing Considerations

- LOCATION and space (TRACKS or RECORDS) must be specified when you are reloading an existing temporary file.
- If you are referencing a file by the DATE parameter and space is given, the space must be equal to the space given when that file was created.
- If you are accessing a file by the LOCATION parameter and space is given, the space must be equal to the space given when that file was created.
- You can create several versions of a file with a program by changing the locations of the files and using different system dates.
- You can create different versions of a file without LOCATION if the space parameters as well as the system dates are different.
- The system assumes that a new file is being created if space is given without LOCATION or DATE and the given filename was found but its space does not match.
- The DATE parameter is only allowed for accessing existing files.
- Whenever a load is performed to an existing file, the system date replaces the previous date for that file.
- If a RETAIN parameter is not specified when an existing file is reloaded, the existing file classification is retained.
- When a scratch file is created, it is not entered in the volume table of contents (VTOC). After the job that created the file is run, the file is lost. The way that an S retain type can appear in the simulation area VTOC is to change a T entry to an S by using RETAIN-S in the FILE statement, or to change a T or P entry to S by using a $DELET SCRATCH statement.

FILE STATEMENT (TAPE)

Function

The FILE statement supplies the system with information about tape files. The system uses this information to read records from and write records to tape.

Placement

You must supply a FILE statement for each new tape file that your program creates, and for each existing tape file that your program uses. (The maximum number of files allowed is explained under *Maximum Number of Files* in the *IBM System/3 Model 12 User's Guide,* GC21-5142.) The FILE statement must follow the LOAD or CALL statement and precede the RUN statement.

Format

// FILE parameters

Content

All parameters are keyword parameters. The parameters are as follows (keywords are in capital letters):

NAME-filename (in program)

UNIT-code

REEL- { name / NL / NS / BLP }

LABEL- { filename (on tape) / 'character string' }

DATE-date

RETAIN-code

BLKL-block length

RECL-record length

RECFM-code (record format)

END-position of tape after processing

DENSITY- { 1600 / 800 / 556 / 200 }

ASCII- { YES / NO }

DEFER- { YES / NO }

CONVERT- { OFF / ON }

TRANSLATE- { OFF / ON }

PARITY- { ODD / EVEN }

The NAME and UNIT parameters are always required. The others are required only under certain conditions.

NAME: The NAME parameter is required. It tells the system the name that your program uses to refer to the file. The NAME parameter must be placed on the first card or line if two or more cards or lines are used for the FILE statement. (See *General Coding Rules* for rules on continuation.)

For the Tape Sort program, you must use specific filenames.

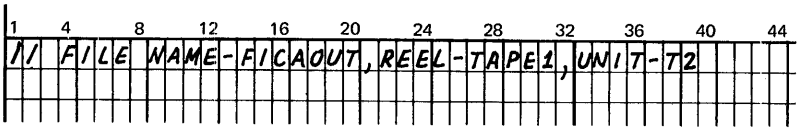
File	Name
Input	INPUT
Output	OUTPUT
Work	WORK1
	WORK2
	WORK3
	WORK4 (optional)

For the Copy/Dump program, you must use specific filenames.

File	Name
Input	COPYIN
Output	COPYO

For the Dump/Restore program, you must use the name BACKUP in the name parameter.

The keyword for the parameter is NAME. It must be followed by the filename used by the program. The first character of the NAME must be alphabetic. The remaining characters can be any combination of characters except commas, apostrophes, or blanks. The number of characters cannot exceed 8. The following example shows how the NAME parameter for a file named FICAOUT would be coded:



UNIT: The UNIT parameter is required. It tells the system the tape unit that contains or will contain the file. The keyword for this parameter is UNIT. It must be followed by a code that indicates the unit. The codes are as follows:

T1	Tape unit 1
T2	Tape unit 2
T3	Tape unit 3
T4	Tape unit 4

The previous example shows how the UNIT parameter would be coded for a file that resides on tape unit 2.

REEL: The REEL parameter is required for tape input files and optional for output files. It identifies the tape that contains or will contain the file. The system uses this parameter to ensure that the correct tape is being used. (For information about how a tape is initialized and identified, see *Tape Initialization* in Part 2 of this manual.)

The REEL parameter can be coded as follows:

REEL-nnnnnn	This format is used for labeled tape volumes. The volume is identified by a code containing a maximum of 6 characters, excluding commas, apostrophes, and blanks. NS, NL, and BLP have special meanings and may not be used as the name of the reel.
REEL-NL	This coding indicates a tape file without a label. The first record of an unlabeled tape must not be an 80-byte record beginning with VOL1.
REEL-NS	This coding indicates an input tape file with a nonstandard label. These labels do not adhere to the IBM Tape Label Standard. The first record of a nonstandard labeled tape must not be an 80-byte record with VOL1 as the first 4 characters. REEL-NS is invalid for output files.
REEL-BLP	This coding is used to bypass label processing on standard labeled tapes. REEL-BLP is invalid for output files.

If the REEL parameter is not specified for an output file, the system assumes the output tape contains standard labels. If REEL-NS, REEL-NL, or REEL-BLP is used, the LABEL, DATE, and RETAIN parameters may not be entered.

Note: User labels are file labels that follow standard header and trailer label conventions (ANSI or IBM). They are a variation of standard labels with a partially fixed format. These labels are sometimes provided by other systems. User labels are not checked by System/3 tape data management and may not be written as part of the label group.

The example under NAME shows how the REEL parameter would be coded for a file on a tape named TAPE1.

LABEL: The LABEL parameter tells the system the name (label) of the tape file as it exists in the header label.

For file creation, the name you supply in the LABEL parameter is used in the header label. If you omit the LABEL parameter, the name from the NAME parameter is used unless REEL-NS, REEL-NL, or REEL-BLP is also specified. Up to 8 characters may be supplied in the LABEL parameter.

For existing files, you must supply the LABEL parameter if the name in the tape label is different from the name your program uses to refer to the file (the NAME parameter). If the header label contains a name longer than 8 characters, only the first 8 characters are recognized by the system for comparison.

The LABEL parameter may not be used with the parameters REEL-NS, REEL-NL, or REEL-BLP. The LABEL parameter can be coded as follows:

LABEL-name	The name entry must begin with an alphabetic character and the remaining characters must not be commas, apostrophes, or blanks.
LABEL-'character string'	A label may also be identified by special characters. The character string must be enclosed in apostrophes, may not contain commas, and is restricted to 8 characters in length. If an apostrophe is used as a character, it must be coded as two apostrophes.

DATE: The DATE parameter tells the system the creation date of an input file. It is used to ensure that the proper version of the file is used. The date specified is compared with the creation date contained in the file label. No comparison is done when DATE is not specified.

For output files, the system date is always used as the creation date. If the DATE parameter is specified for an output file, the system compares the specified date with the creation date of the file already on the tape. If no file exists on the tape, or a file with a different label exists, or the dates do not agree, the system halts.

The date may be coded in one of two formats: month-day-year (mmddyy), or day-month-year (ddmmyy). The format must match the format of the system date chosen at system generation time.

The DATE parameter may not be specified with REEL-NS, REEL-NL, or REEL-BLP.

RETAIN: The RETAIN parameter specifies the number of days a file should be retained before it expires. This number may be from 0 to 999. After the number of days has elapsed, the file expires and the system allows the file to be written over. If the RETAIN parameter is omitted, a value of zero is assumed. A value of 999 indicates a non-expiring permanent tape file.

If an attempt is made to write over an unexpired file, the system halts, allowing the operator to cancel the job or continue. A tape containing a permanent tape file must be reinitialized before it can be used for output. The RETAIN parameter may not be used with REEL-NS, REEL-NL, or REEL-BLP.

BLKL: The BLKL (block length) parameter specifies the number of bytes in a physical block on tape. The block length can be from 18 bytes to 32,767 bytes. The maximum length is limited to the main storage not occupied by the program and supervisor. The block length must be an integral multiple of the record length for fixed (F) and fixed blocked (FB) files (see RECFM parameter). If an ASCII file is being used, any existing block prefixes must be included in the block length.

RECL: The RECL (record length) parameter specifies the number of bytes in a logical record. The maximum record length is 32,767 bytes. The minimum record length permitted for F and FB type files is 18 bytes (see RECFM parameter). The record length for V, VB, D, and DB type files must include the 4-byte record descriptor.

RECFM: The RECFM (record format) parameter identifies the format of the input or output file records. The parameter entries are:

F Fixed length, unblocked records. Logical and physical records are the same size.

V Variable length, unblocked records. Each physical record contains one logical record; the logical record can vary in length.

D Variable length, unblocked records in the D-type ASCII format.

FB Fixed length, blocked records. All records are of equal length and all blocks are of equal length. Each physical record contains more than one logical record.

VB Variable length, blocked records. Each physical record contains logical records of various lengths.

DB Variable length, blocked records in the D-type ASCII format.

END: The END parameter specifies the position of the tape after the file has been processed. The options are as follows:

LEAVE The tape remains in the position it was in after the last record was read or written.

REWIND The tape is rewound to the load point.

UNLOAD The tape is rewound and unloaded for removal from the tape drive.

If the END parameter is omitted, REWIND is assumed.

DENSITY: The DENSITY parameter is used to specify the number of bpi (bits per inch) at which files are to be written or read. The parameter must specify the density at which the tape was initialized. See $TINIT (Tape Initialization Program) description in this manual. For 9-track tapes this parameter affects only the density of nonlabeled output files. When standard labeled or nonstandard labeled tapes are used, the tape hardware automatically determines the density at which the tape was initialized. When a tape is initialized at 1,600 bpi with standard labels, any file that is written on that tape is at 1,600 bpi, regardless of the parameter specified for DENSITY. No error halts occur if an incorrect 9-track density is specified. The parameter entries are:

1600 The file is to be written at 1,600 bpi (valid for all 9-track tape units).

800 The file is to be written or read at 800 bpi (valid for 9-track dual density tape units or for all 7-track tape units).

556 The file is to be written or read at 556 bpi (valid for all 7-track tape units).

200 The file is to be written or read at 200 bpi (valid for all 7-track tape units).

If the DENSITY parameter is omitted, 1,600 bpi is assumed on 9-track tape units, and 800 bpi is assumed on 7-track tape units.

ASCII: The ASCII parameter (ASCII-YES or ASCII-NO) is used to indicate to the system when an ASCII file is being used. If ASCII files are being used, ASCII-YES must be coded. ASCII-YES is invalid for files on 7-track tape units. If this parameter is omitted or coded ASCII-NO, an EBCDIC file is assumed.

DEFER: The DEFER parameter (DEFER-YES or DEFER-NO) tells the system whether the file will be mounted on a tape drive when the file is allocated and opened. If the tape volume is not online, DEFER-YES must be coded. If the parameter is omitted, DEFER-NO is assumed.

Note: For RPG II object programs, this option should only be used for files that use the same drive as a table file. All other files are allocated and opened at the beginning of the program.

Other programs (such as COBOL object programs) that do not allocate and open all files at the same time, or that do so conditionally by program logic, should not use the DEFER-YES option.

DEFER-YES cannot be used if BSCA or devices attached to SIOC are used in the program.

Multivolume Tape Files

The FILE statement for processing multivolume tape files requires that you define and code the UNIT and REEL parameters differently than you would for single-volume files. There are two reasons for this:

- When processing tape files contained on more than a single volume, the system requires information about each volume in order to perform all the checking and protection functions necessary.
- Additional information is needed to determine and check the sequence in which the volumes are processed and when they are to be mounted on the tape drives.

When an end-of-volume condition is reached on a multivolume file, that volume rewinds to load point and unload. The message 'EOV Tn' is printed if LOG is on (where n = 1, 2, 3 or 4). If the drive that is to contain the next volume (whether the same drive or another drive), is not in a ready condition, the system comes to I/O attention. Processing continues when the drive that is to contain the next volume is made ready. If you are using alternating drives, and the next volume is mounted and the drive is ready when end of volume is reached, the message is printed and processing continues without stopping.

For multivolume tape files, the UNIT and REEL parameters of the FILE statement may require a list of codes. The following rules apply:

- The list must be enclosed by apostrophes.
- The items in the list must be separated by commas.
- Nine- and seven-track units cannot be intermixed.

The considerations for coding multivolume parameters are included in the following parameter discussions. The functions of the parameters are explained under *FILE Statement (Tape).* Parameters not mentioned here are used as explained under *FILE Statement (Tape).*

Note: Multivolume tape files cannot be used if BSCA or other interruptible devices are used in the program.

The maximum number of multivolume files allowed is explained under *Maximum Number of Files* in *IBM System/3 Model 12 User's Guide,* GC21-5142.

REEL: The names of the tapes that contain or will contain the multivolume file must follow the keyword REEL (40 names maximum). If the input tapes are not labeled, the REEL parameter must be coded REEL-'NL,n'; if the input tapes contain nonstandard labels, the REEL parameter must be coded REEL-'NS,n'. If the input tapes have standard labels, and label processing is to be bypassed, the code is REEL-'BLP,n'. The n in each case is the number of volumes in the file (99 volumes maximum). For output files, the n in REEL-'NL,n' is ignored.

UNIT: The keyword UNIT must be followed by a code or codes indicating the location of the tape unit that contains or will contain the file. No UNIT parameter may be repeated. The order of codes in the UNIT parameter must correspond to the order of names in the REEL parameter. When the number of codes in the UNIT parameter is less than the number of codes in the REEL parameter, the units are used alternately.

In the following example, line (A) shows a tape multi-volume file consisting of three reels. The volumes must be mounted as follows:

INVREEL1 on tape unit T1

INVREEL2 on tape unit T2

INVREEL3 on tape unit T3

Line (B) shows a three-volume file with nonstandard tape labels. The volumes must be mounted as follows:

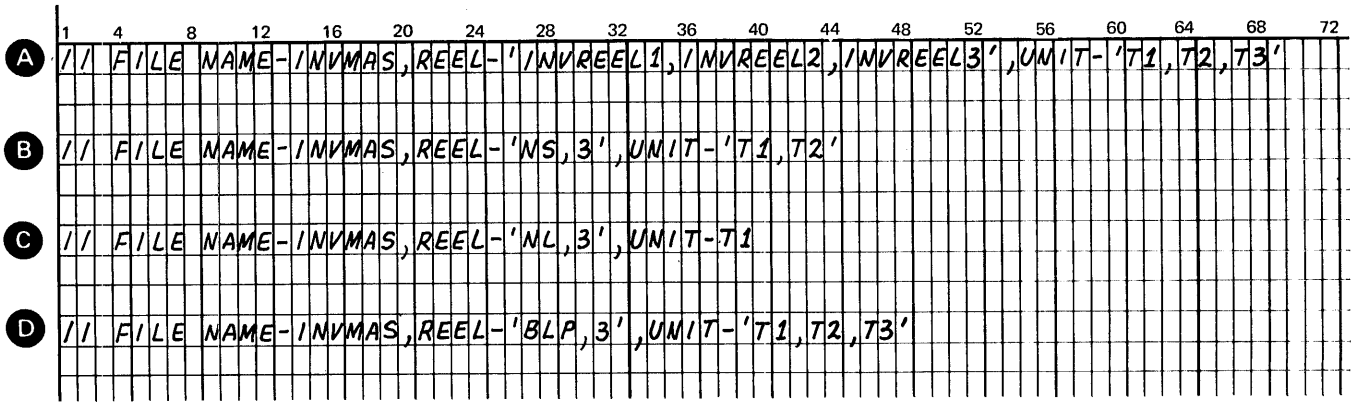
First volume on tape unit T1

Second volume on tape unit T2

Third volume on tape unit T1

Line (C) shows a three-volume file with unlabeled reels. The volumes must be mounted in sequence on tape unit T1.

Line (D) shows the three-volume standard labeled file of line (A) with label processing bypassed.



FORMS STATEMENT

See *PRINTER Statement.*

HALT STATEMENT

Function

The HALT statement tells the system to halt when a program ends. The operator can restart the system when he is ready, and the system continues reading the next OCL statements.

The HALT statement is needed only if you want to cancel the effect of a NOHALT statement.

Placement

A HALT statement can be placed anywhere among the OCL statements. In a procedure it must precede the RUN statement.

Format

// HALT

Content

None (comments may be entered starting in column 9).

IMAGE STATEMENT

Function

To operate correctly, the printer requires characters matching those on the printer chain to be in a special area of core storage called the chain-image area. When you replace the printer chain with one having different characters, you must also change the contents of the chain-image area.

The IMAGE statement instructs the system to replace the contents of the chain-image area with the characters indicated by the statement. The characters can be entered from the system input device or from a source library. The effect of the IMAGE statement is temporary, and the system chain-image is returned to the chain-image area when IPL occurs.

Placement

The IMAGE statement can appear anywhere among the OCL statements. In a procedure, it must precede the RUN statement.

Format

// IMAGE parameters

Content

The IMAGE statement tells the system either of two things: (1) the new chain characters are to be read from the system input device; or (2) the new chain characters are to be read from the source library.

The IMAGE parameters are:

format-*HEX, CHAR, or MEM*

number-value

name-name

unit-code

(Coding only HEX, CHAR, or MEM is preferable for format but HEXADECIMAL, CHARACTER, or MEMBER can be coded.)

Characters From the System Input Device

If you wish to indicate that the new chain characters are to be read from the system input device, use the following parameters:

Format: Use the word CHAR to indicate that the characters are in EBCDIC form. Use the word HEX to indicate that the characters are in hexadecimal form.

Number: The number parameter must be used with HEX and CHAR. It must be a value that is equal to the number of columns or line positions in the data statements or the data keyed in following the IMAGE statement that contains the new characters. This number must not exceed 240 when the characters are hexadecimal, 120 when characters are EBCDIC. The name and unit parameters must not be coded.

Following are the rules for punching or keying the new characters:

- The characters must begin in column or line position 1.
- Consecutive card columns or line positions must be used; however, only the first 80 columns or line positions of the card or line can be used. Hexadecimal requires an even number of columns or line positions, two per character.
- To continue the characters on another card or line, begin the characters in column or line position 1.

Characters from Source Library

To indicate that new chain characters are to be read from the source library on disk, the format parameter must specify the word MEM.

The following parameters must also be included:

Name: The name parameter identifies the source member containing the characters in the library. The only way you can place the characters in a source library is by using the Library Maintenance program. The name you supply in Library Maintenance control statements is the name used to identify the characters in the source library.

Unit: The unit parameter must be used with the name parameter. It is used to indicate which simulation area on disk contains the source library. The codes used are R1, F1, R2, and F2.

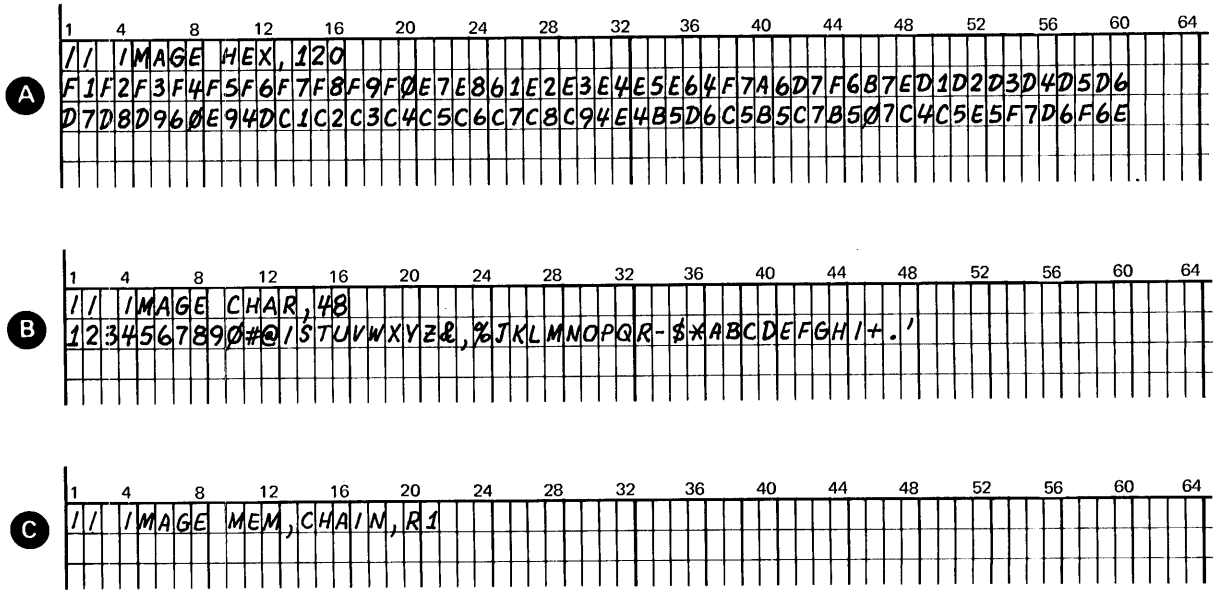
Example

The IMAGE statement in example (A) tells the system that the new characters are on data statements or keyed in. The format parameter indicates that new characters are in hexadecimal form; the number parameter indicates that there are 120 columns or line positions containing the new characters.

In example (B), the new characters, on data statements or keyed in, are in EBCDIC. The number parameter indicates that there are 48 columns or line positions containing the new characters.

Example (C) tells the system that the new characters are to be read from the source library. The format parameter indicates that the new chain characters are in the source library. The name parameter indicates that the characters were named CHAIN in the source library. The unit parameter indicates that the source library containing them is in the simulation area R1 on drive 1. Examples of the member specified in example (C) are the data portions of examples (A) and (B). The member itself requires a // IMAGE statement with the characters either in hexadecimal or EBCDIC. The number of columns or line positions containing the characters must also be specified.

(See *Library Maintenance* in Part 2 for restrictions on the name used in coding MEM.)



JOB STATEMENT

Function

The JOB statement provides the user with the following functions:

- Allows the user to group related jobs in the spool file by identifying the group with a common groupname. Each job is further identified in the spool file by the jobname from the LOAD statement. If the jobname is not supplied, the program name further identifies the job.
- Allows the user to specify whether jobs following the JOB statement are to be spooled.
- Allows the user to assign priority to jobs in the spool file. Jobs contained in the spool file are scheduled for printing in the order of their priority.

Placement

The JOB statement precedes the first LOAD or CALL statement for a group. It cannot be used in a procedure. When a rollin is pending for a program level, a JOB statement read by that level will be ignored.

Format

//groupname JOB parameters

Content

groupname: This is a required entry used to uniquely identify a group of jobs in the spool file under the same name. Groupname may not exceed 8 characters in length or contain embedded blanks or commas. Groupname should contain only characters that are on the 5471 keyboard when the 5471 is attached to the system. All keyword parameters are optional on the JOB statement. When more than one keyword parameter is specified, they must be separated by commas.

SPOOL: The SPOOL parameter is used to specify whether jobs are to be spooled. SPOOL-NO specifies that jobs following the JOB statement are not to be spooled; consequently, print requests from these jobs will not be intercepted. SPOOL-YES indicates that print requests from the jobs following the JOB statement are to be intercepted by spool. If the SPOOL parameter is not specified, SPOOL-YES is assumed.

PRIORITY: A priority may be assigned to a job to indicate its level of importance in the spool file. The priority of the job in the spool file is that priority assigned by the JOB statement. A priority of 0 causes a job to be put in the spool file in a hold state with a priority of 1. The job put on hold may be released via an operator control command. (See *IBM System/3 Model 12 User's Guide,* GC21-5142, for a list of operator control commands.) Priority 5 is the highest priority that may be assigned. Within a given priority, jobs are scheduled on a first-in, first-out basis. If this parameter is not specified, priority 1 is assumed.

Note: When keyword parameters are not specified on this statement, comments may not be given following the JOB statement identifier.

LOAD AND LOAD * STATEMENT

Function

The LOAD statement identifies the program to be executed and indicates whether the program is to be loaded from the system input device for the program level or from an object library.

Placement

One LOAD statement is required for each program executed. The only requirement is that the LOAD statement precede the RUN statement.

Format

The LOAD statement has two formats:

//jobname LOAD * (a blank is mandatory between LOAD and *)

//jobname LOAD program-name,unit

The first format is used to load object programs from the system input device. The second format is used to load object programs from the object library.

Content

jobname: This optional entry is used to uniquely identify a job. If specified, the jobname must begin in position 3 of the statement, must not exceed 8 characters in length, and may not contain commas, apostrophes, periods, or blanks. Jobnames should contain only characters that are on the 5471 keyboard when the 5471 is attached to the system.

If no jobname is specified, the system assigns one. If the jobname is assigned by the system, it is made up of the program name from the LOAD statement and a two-digit number assigned by the system. Jobnames assigned by the system are incremented by one at the end of the job in which a jobname is assigned. If a LOAD * statement without a jobname is encountered, the system assigns a jobname of ASTRSKnn. The number portion of the jobname is reset to 01 whenever a JOB statement is encountered. After 99 jobnames have been assigned within one group, the number is reset to 01. When the print queue is displayed, the jobname identifies jobs on the queue.

Asterisk: An asterisk is specified when the user wants the object program loaded from the program level's system input device. The object program must follow the RUN statement for the program. A /* statement must follow the object program to indicate the end of the object program input. The object program is temporarily copied into the object library on the system pack and then loaded into main storage for execution. Only level 1 may contain a LOAD * program.

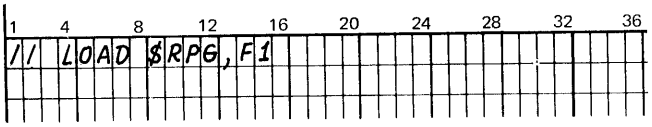
program-name: The program-name is the name used to identify the program in the object library on disk and may be up to 6 characters in length. The name must begin with one of 29 characters (A-Z, @, #, or $) and may be followed by up to 5 additional characters. Commas, apostrophes, periods, and blanks may not be used in the program-name. The system utility programs and program products are identified by the following names:

Program	Name
Alternate Track Assignment	$ALT
Alternate Track Rebuild	$BUILD
Assembler	$ASSEM
COBOL	$CBL00
Copy/Dump	$COPY
Disk Initialization	$INIT
Disk Sort	$DSORT
Dump Restore	$DCOPY
File and Volume Label Display	$LABEL
File Delete	$DELET
FORTRAN	$FORT
GANGPUNCH	$GANGP
Library Maintenance	$MAINT
List	$CLIST
Macro Processor	$MPXDV
MFCU Sort/Collate	$CSORT
Multileaving Remote Job Entry	$$MRJE
Overlay Linkage Editor	$OLINK
Reassign Alternate Track	$RSALT
Reproduce and Interpret	$REPRO
Remote Job Entry	$$RJE
Restart	$$RSTR
RPG Linkage Editor	$LINKB
RPG II Auto Report	$AUTO
RPG II Compiler	$RPG
Simulation Area	$SCOPY
Spool Writer	$SSWTR
Tape Initialization	$TINIT
Tape Sort	$TSORT
Tape Error Summary	$TVES
1255 Utility	$MICR

unit: The unit parameter is a code indicating which simulation area on disk contains the program. The codes are R1, F1, R2, and F2. The unit parameter is required because your programs can be on any of the simulation areas. The disk area containing your object program is called an object library. You can create an object library in any of the simulation areas (R1, F1, R2, F2) by using the Library Maintenance program. (See *Library Maintenance* in Part 2 of this manual.)

Example

In the following sample LOAD statement, $RPG is the name that identifies the RPG II Compiler.



F1 is the code indicating the simulation area on drive 1 where the compiler is located in this example.

LOCKOUT STATEMENT

Function

The LOCKOUT statement is used only in DPF systems. It is used to suspend the other program level to allow fast job initiation in the program level in which it is entered. Job initiation is slowed if both program levels use the system transient area and/or disk drive 1. The other program level remains suspended until job initiation is complete.

Note: This statement should not be used when the active program level is using time-dependent devices such as BSCA and serial I/O channel.

Placement

The LOCKOUT statement can be placed anywhere among the OCL statements, but must precede the RUN statement.

Format

// LOCKOUT

Content

None (comments may be entered starting after the first blank column).

LOG STATEMENT

Function

OCL statements and message codes are printed on the printer-keyboard. If your system has no printer-keyboard, the statements and codes are printed on the printer. The device used to print OCL statements and message codes is called the *logging device.* If you want to change the logging device, or specify whether or not the statements and codes are to be printed, you must use a LOG statement.

In a DPF programming environment, the same logging device is always used in both program levels. Hence, if the logging device is changed in one program level, the new logging device will also be used by the other program level.

The LOG statement tells the system to do one of four things:

- Use the printer as the logging device
- Use the printer-keyboard as the logging device
- Stop printing OCL statements and message codes
- Start printing OCL statements and message codes

In addition it may tell the system whether or not to suppress eject at end of job.

Placement

You can use the LOG statement within any of the sets of OCL statements for your programs. In a procedure it must precede the RUN statement.

Format

// LOG code,mode

Content

The following four codes and two modes can be used as parameters:

Code	Meaning
CONSOLE	Use printer-keyboard as logging device
PRINTER	Use printer as logging device
OFF	Stop logging
ON	Start logging

Mode	Meaning
EJECT	Eject a page at end of job
NOEJECT	Suppress page eject at end of job

Only one code and one mode can be used in each LOG statement. The start of logging is assumed if CONSOLE or PRINTER is specified.

When the system reads a LOG statement that contains the OFF code, it stops printing OCL statements and message codes. The only way you can instruct the system to start printing them again is by using a LOG statement that contains ON, PRINTER, or CONSOLE. When ON is specified, printing resumes on the last logging device specified. However, the system suspends logging during the time that the log device (excluding the 5471) is allocated to a program in either program level. Logging resumes when the program using the log device goes to end of job.

The NOEJECT mode is used to stop the page eject at end of job. If neither EJECT or NOEJECT is specified, EJECT is assumed. NOEJECT stays in effect until a LOG statement without NOEJECT is read or until an IPL is performed. EJECT stays in effect until a LOG statement with NOEJECT is read. EJECT is only active when logging to the printer.

NOHALT STATEMENT

Function

Normally the system halts when a program ends. The NOHALT statement tells the system to read the next set of OCL statements without stopping. The effect of this statement lasts until the system reads a HALT statement or an IPL occurs. Under certain conditions, the effect of the NOHALT statement is ignored temporarily when an abnormal end of job occurs. The system reverts to the NOHALT mode after a response.

Placement

A NOHALT statement can be placed anywhere among the OCL statements. In a procedure it must precede the RUN statement. The NOHALT statement can be submitted in program level 1 or 2.

Format

// NOHALT SEVERITY-code

Content

SEVERITY: This parameter indicates the severity code of halts that the system is allowed to select default options for. If the SEVERITY parameter is not specified, the operator must respond to all halts except EJs. The code must be one of the following: 1, 2, 4, or 8. If the severity assigned to a system halt is greater than the severity indicated in the NOHALT statement, the system halts and waits for the operator's response. If the severity assigned to the halt is equal to or less than the severity indicated in the NOHALT statement, the system selects the default option for the halt and processing continues. The severity code does not affect system halts having no default options. Operator intervention is required in those cases.

Severity code 1 is the least severe; severity code 8 is the most severe. In most cases the default option is ignored when system halts cannot be printed or spooled. In this case the operator must respond to the halt.

Note: Some halts are defaulted when the system is using the // NOHALT SEVERITY code statement. When using spooling and the print writer is active, the system halts with a SPPPEH halt after the print queue is empty. If the operator responds with a 0 option, the print writer continues to search the print queue, and the SPPPEH halt is defaulted until the print writer has started and finished printing the next job put in the print queue.

PARTITION STATEMENT

Function

The PARTITION statement is used only in DPF systems and guarantees a minimum size to program level 2 for a program in that level.

Placement

The PARTITION statement can be placed anywhere among the OCL statements preceding the RUN statement. The PARTITION statement cannot be submitted in program level 2.

Format

// PARTITION size

Content

Size: The size parameter specifies the number of bytes of storage needed for program level 2. (See *Loading Programs in a DPF Environment* in *IBM System/3 Model 12 User's Guide,* GC21-5142.)

PAUSE STATEMENT

Function

The PAUSE statement causes a halt. It usually is used to give the operator time to prepare for the next program. He might, for example, have to place a data module on drive 2. Comment statements that give the operator instructions usually precede PAUSE statements.

When the operator is ready, he can restart the system. The system continues reading the OCL statements that follow the PAUSE statement.

Placement

PAUSE statements can be placed anywhere among the OCL statements. A // PAUSE statement prior to a // LOAD statement (between jobs) causes a 90 halt with a continue option (recovery 0) only. A // PAUSE placed between the // LOAD and // RUN statements (within the OCL sequence) causes a 91 halt with a continue (recovery 0) or a cancel (recovery 2 or 3) option.

Format

// PAUSE

Content

None (comments may be entered starting in column 10).

PRINTER STATEMENT

Function

The PRINTER statement allows you to define the system print device and control options related to print spooling. The FORMS statement identifier may be used in place of the PRINTER statement identifier.

Placement

The PRINTER statement can be placed anywhere among the OCL statements. In a procedure it must precede the RUN statement.

Format

// PRINTER parameters

Content

The parameters are as follows (keywords are in capital letters; defaults are underlined):

DEVICE- { 5203 / <u>5203L</u> / 5203R }

LINES-nnn

FORMSNO-nnn

COPIES-nn

DEFER- { <u>YES</u> / NO }

ALIGN- { YES / <u>NO</u> }

DEVICE: The DEVICE parameter is optional, but if it is specified it must be followed by the name of the print device. For an IBM 1403 Printer or a single-carriage IBM 5203 Printer, either 5203 or 5203L is a valid device name. For a dual-carriage IBM 5203 Printer, either 5203 or 5203L specifies the left carriage and 5203R specifies the right carriage. You may omit the DEVICE parameter entirely (default parameter is 5203L, left carriage).

LINES: The LINES parameter is optional. It is used to alter the number of print lines (forms length) per page. Possible range is 1 to 112. However, some system utility programs require a minimum of 12. The number of lines specified remains in effect for that level until another PRINTER statement with LINES parameter is entered or until the next IPL. This parameter overrides the forms length specified during system generation; however, a program's forms length overrides the LINES parameter. If a program's forms length is used, it is in effect for the duration of that job only. At the end of the job, forms length is restored to the previous value.

FORMSNO: This optional parameter may be used to tell the operator which forms are to be mounted on the printer. This parameter can be any combination of 1 to 3 characters, except commas, apostrophes, or embedded blanks. When this parameter is used and spool is not intercepting print requests, the system halts with a CR8LLT (mount forms on left carriage) or CR8LRT (mount forms on right carriage) halt. When printing spooled printed output, the print writer issues a message whenever the forms type for the current print job is different from that of the previous print job. The response taken to this message tells the print writer whether separator pages should be printed between jobs. See the *IBM System/3 Model 12 Operator's Guide,* GC21-5144, for information on separator pages. The FORMSNO parameter applies only to the job in which the PRINTER statement is received.

COPIES: This optional parameter allows the user to obtain from 1 to 99 copies of a job's spooled printed output. If this parameter is not specified, only one copy is printed. When more than one copy is requested, the print writer continues to produce the number of requested copies before continuing to the next job. This parameter is ignored when print spooling is inactive or not supported for the specified device. The COPIES parameter applies only to the job in which the PRINTER statement is received.

DEFER: The DEFER parameter is optional. It is ignored when print spooling is inactive or not supported for the specified device. DEFER-NO allows the spooling user to begin printing a job's printed output before the job has completed execution if the job is the next job to be printed from the print queue. When DEFER-YES is specified, printing does not begin until the job has completed execution. The DEFER parameter applies only to the job in which the PRINTER statement was received. If the parameter is not specified, the system assumes DEFER-YES.

ALIGN: The ALIGN parameter is optional. It aids the operator in forms alignment for spooled printed output. This parameter is ignored when print spooling is inactive or not supported for the specified device. When ALIGN-YES is specified, the printer stops after printing the first line to allow forms alignment. A halt is displayed on the message display unit after the first line is printed. The operator's response to this message indicates that forms are aligned (continue printing) or that the line should be printed again (try alignment again). If more than one copy is requested (COPIES parameter) and ALIGN-YES is specified, the printer halts for forms alignment prior to printing each copy. If ALIGN-NO is specified, the printer does not stop. The ALIGN parameter applies only to the job in which the PRINTER statement was received. If the parameter is not specified, the system assumes ALIGN-NO.

Note: If logging was assigned to the 1403/5203, forms alignment is done on the first OCL statement logged for that job. For this reason, logging to the 1403/5203 should be suppressed when ALIGN-YES is used.

Spooling Considerations: When a PRINTER statement is encountered and printer output for the job is being spooled, the effect of the COPIES, DEFER, ALIGN, and/or FORMSNO parameters is delayed until the print writer is ready to print the output.

PUNCH STATEMENT

Function

The PUNCH statement enables you to change the system punch device.

Placement

The PUNCH statement can be placed anywhere among the OCL statements. In a procedure it must precede the RUN statement.

Format

// PUNCH code

Content

The codes that can be used as parameters are:

Code	Meaning
MFCU1	Primary hopper of the MFCU
MFCU2	Secondary hopper of the MFCU
1442	Card read/punch
3741	Data station (96-byte records) or programmable work station

READER STATEMENT

Function

The device used to read OCL statements is called the *system input device.* The READER statement assigns the system input device to the device specified.

Placement

The READER statement must not come between the LOAD or CALL statement and a RUN statement. If a READER statement is used in a procedure, the system input device is changed when the READER statement is processed; OCL statements are not read from the new system input device until the procedure is completely executed. If you use the READER statement to change the system input device, the device you specify is used to read source programs, control statements, or OCL statements. Changing the system input device affects the placement of source programs and control statements as well as OCL statements.

Format

// READER code

Content

The codes are:

Code	Meaning
CONSOLE	Printer-keyboard
MFCU2	Secondary hopper of the MFCU
MFCU1	Primary hopper of the MFCU
1442	Card read/punch
3741	Data station (96-byte record) or programmable work station

RUN STATEMENT

Function

The RUN statement indicates the end of the OCL statements for a program. After the system reads the RUN statement, it runs the program or merges the procedure into the job stream.

Placement

A RUN statement is needed for each of the programs you want the system to run. In the job stream, it must be the last statement within each of the sets of OCL statements for your programs. It can also be the last OCL statement in a procedure. (For more information about procedures, see *Procedures* in *IBM System/3 Model 12 User's Guide,* GC21-5142.)

Format

```
// RUN
```

Content

None (comments may be entered starting in column 8).

SIMULATE STATEMENT

Function

The SIMULATE statement is used to enable and disable simulation areas R2 and F2 on drive 2. R2 and F2 are simulated on D2 at IPL. To allow processing multivolume files on D2, simulation must be turned off on D2.

Placement

The SIMULATE statement must not come between a LOAD or CALL and a RUN statement. It cannot be used in a procedure. It is invalid if the other level is in a nested procedure or is not at end of job, or if rollin is pending. It is invalid to turn simulation off if spool is using D2.

Format

// SIMULATE $\frac{\text{ON}}{\text{OFF}}$

Content

ON specifies simulation turned on for D2. This enables R2 and F2. Simulation on D2 remains on until a // SIMULATE OFF statement is read. OFF specifies simulation turned off for D2. This disables R2 and F2. Simulation for D2 remains off until IPL or a // SIMULATE ON statement is read.

SWITCH STATEMENT

Function

The SWITCH statement sets one or more external indicators on or off. The indicators are always off after the operator uses the IPL procedure to start the system. If a SWITCH statement is used to set an indicator on, the indicator remains on until another SWITCH statement sets it off, or until the operator again uses the IPL procedure to start the system. There can be only one SWITCH statement per job.

Placement

The SWITCH statement can appear within any of the sets of statements for your programs. The only requirements for the SWITCH statement are that it must follow the LOAD or CALL statement and precede the RUN statement.

Format

// SWITCH indicator-settings

Content

Indicator-settings: The indicator-settings parameter is a code that consists of 8 characters, one for each of the eight external indicators (U1-U8). The first, or leftmost, character gives the setting of indicator U1; the second character gives the setting of U2; and so on.

The code must always contain 8 characters. For each indicator, one of the following characters must be used:

Character	Meaning
0	Set the indicator off
1	Set the indicator on
X	Leave the indicator as it is

Example

The code 1X0110XX would cause the following results:

Indicator	Result
U1	Set on
U2	Unaffected
U3	Set off
U4	Set on
U5	Set on
U6	Set off
U7	Unaffected
U8	Unaffected

/& STATEMENT

Function

/& statements are used as a precautionary measure. Placed in front of your OCL set, a /& statement signals the system that a new set of OCL statements is coming. It prevents your statements from being read as a part of the preceding set of statements or data. Any attempt to read more data from that device will be blocked.

Placement

/& statements are not required. It is recommended, however, that you use them as the first statement in each of the sets of OCL statements for your programs. They are not allowed in a procedure.

Format

/&

Content

None (comments may be entered starting in column 4).

/* STATEMENT

Function

/* statements are not true OCL statements, but are used to indicate the end of a data file read in from a card reader, console, or 3741.

Placement

A /* statement should be the last statement of an input data file or program deck.

Format

/*

Content

None (comments may be entered starting in column 4).

*(COMMENT) STATEMENTS

Function

Comment statements are commonly used either to explain the jobs or to give the operator instructions. Operator instructions are usually given in connection with a PAUSE statement. Comment statements are printed along with the other OCL statements. They have no other effect on the system.

Placement

In OCL statements, you can include special statements that contain only comments. Comment statements must contain an asterisk (*) in column 1. They can be placed anywhere among the OCL statements in either a job stream or a procedure.

Format

*comment

Content

The comment can be any combination of words and characters. The only requirement is that an asterisk (*) be in column 1.

Introduction to System Utility Programs

The Model 12 SCP includes a group of system utility programs that are resident in a simulation area. These programs perform a variety of functions, such as preparing data modules for use, reorganizing an indexed file, and deleting files. Each of the system utility programs is described separately in this section, with the following information given for each program:

- Functions performed
- OCL statements required to use the program
- Parameter explanations
- OCL (operation control language) considerations
- Examples

OCL STATEMENTS

Each system utility program requires a set of OCL statements. The first statement required within a set of OCL statements is the LOAD statement. It identifies the program to be run and indicates which simulation area the program will be loaded from. The statement format used to load a program from the simulation area is:

```
// LOAD program-name,unit
```

The program-name in the LOAD statement specifies the system utility program you want to run. The following list contains the system utility programs described in this section, the name that must appear on the LOAD statement, and the main storage requirements for selected SCP programs (size is the minimum main storage, in bytes, excluding supervisor requirements):

Program	Name	Size
Tape Initialization	$TINIT	8K
Tape Error Summary	$TVES	8K
Disk Initialization	$INIT	8K
Alternate Track Assignment	$ALT	8K
Alternate Track Rebuild	$BUILD	8K
File and Volume Label Display	$LABEL	8K to 18K (note 3)
File Delete	$DELET	8K
Copy/Dump	$COPY	8K (note 1)
Dump/Restore	$DCOPY	8K
Simulation Area	$SCOPY	8K
Library Maintenance	$MAINT	8K (notes 1 and 2)
Reassign Alternate Track	$RSALT	8K

Notes:

1. Uses more main storage, if available.
2. Requires a dedicated system (cannot be used with dual programming).
3.

50	File VTOC	8K
1000	File VTOC	10K
1-1000	Entries unsorted	10K
1-300	Entries sorted	10K
301-500	Entries sorted	12K
501-700	Entries sorted	14K
701-900	Entries sorted	16K
901-1000	Entries sorted	18K

The unit parameter specifies a code that describes the location of the simulation area which contains the system utility program. The codes are F1, R1, F2, and R2.

The RUN statement also is required for each system utility program. The format of this statement is:

```
// RUN
```

The program begins after the system reads this statement. One or more FILE statements may be required, depending on the system utility program to be run and the function to be performed. (See the following system utility program descriptions for FILE statement requirements.)

CONTROL STATEMENTS

All of the programs require utility control statements (except $TVES), which you must supply. These statements give the program information concerning the output you want the program to produce or the way in which you want the program to perform its function. The programs read these statements from the system input device or a procedure. They must be the first input read by the programs.

Every control statement is made up of an *identifier* and *parameters.* The identifier is a word that identifies the control statement. It is always the first word of the statement. Parameters are information you are supplying to the program. Every parameter consists of a keyword, which identifies the parameter, followed by the information you are supplying.

Coding Rules

The rules for constructing control statements are as follows:

1. *Statement identifier.* // followed by a blank should precede the statement identifier. Do not use blanks within the identifier.

2. *Blanks.* Use one or more blanks between the identifier and the first parameter. Do not use them anywhere else in the statement.

3. *Statement parameters.* Parameters can be in any order. Use a comma to separate one parameter from another. Use a hyphen (-) within each parameter to separate the keyword from the information you supply. Do not use blanks within or between parameters.

4. *Statement parameters containing a list of data after the keyword.* Use apostrophes (') to enclose the items in the list. Use a comma to separate one item from another. For example: UNIT-'R1,R2' (R1 and R2 are the items in the list).

5. *Statement length.* All control statements except disk initialization, simulation area, and library maintenance statements must not exceed 96 characters. The following library maintenance statements can be continued on another statement. (See *Continuation* under *Coding Rules* in Part 1 of this manual.)

   // ALLOCATE

   // COPY (not COPY statements read from a file or ENTRY statements)

   // DELETE

   // MODIFY (not REMOVE, REPLACE, or INSERT statements)

   // RENAME

The disk initialization statement // VOL can also be continued. All simulation area control statements may be continued.

The following is an example of a control statement:

```
// COPY FROM-F1,LIBRARY-O,NAME-SYSTEM,TO-R1
```

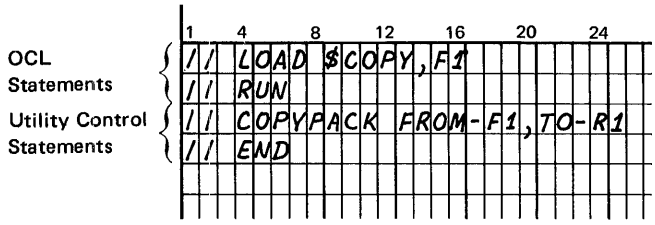
The statement identifier is COPY. The parameter keywords are FROM, LIBRARY, NAME, and TO. The information you supply is F1, O, SYSTEM, and R1.

END Control Statement

The END statement is a special control statement that indicates the end of control statements. It consists of // END starting in position 1 and must always be the last control statement for the program (except $TVES).

Placement of Control Statements in the Job Stream

Control statements for utility programs must follow the RUN statement. The following example shows the use and placement of utility control statements.



Special Meaning of Capital Letters, Numbers, and Special Characters

Capitalized words and letters, numbers, and special characters have special meanings in OCL and utility control statement descriptions.

In utility control statements, capitalized words and letters must be written as they appear in the statement description. Sometimes numbers appear with the capitalized information. These numbers must also be written as shown.

Words or letters that are not capitalized mean you must use a value that applies to the job you are doing. The values that can be used are listed in the parameter summaries for the control statements. For example, ID-xxxxxx means you must supply the value where the xxxxxx appears.

Braces and brackets ([{ }]) sometimes appear in parameters shown in control statement summaries and parameter summaries. They are not part of the parameters. Braces indicate that you must choose one of several values to complete the parameter. For example, RETAIN-{T/P} means you can use either RETAIN-T or RETAIN-P. Brackets indicate optional parameters. For example, [,PACK-name] is an optional parameter that may or may not be used.

Tape Initialization Program—$TINIT

The Tape Initialization program prepares tapes for use. It writes IBM standard volume labels on tape so that tape data management can perform IBM standard label processing. The program is available on either card or disk.

The Tape Initialization program performs these functions at your request:

- CHECK labeled tapes for a volume label and an unexpired file before writing a new volume label.
- CLEAR labeled or unlabeled tapes by bypassing CHECK and unconditionally initializing the tape.
- DISPLAY the volume and header labels.

All tapes must be initialized before use. Tapes that have been initialized need not be reinitialized unless you want to write a new volume label or use a tape that contains a permanent file for output. This program can either initialize (CLEAR or CHECK) or DISPLAY one tape per unit during the same program run.

CONTROL STATEMENT SUMMARY

Use	Control Statement

Check for an expired file and a label, then write a new label.	// VOL UNIT-{T1 / T2 / T3 / T4},REEL-{NL / xxxxxx},TYPE-CHECK,ASCII-{YES / NO},
	DENSITY-{1600 / 800 / 556 / 200},ID-yy...yy
	// END

Write volume label without checking for old label.	// VOL UNIT-{T1 / T2 / T3 / T4},REEL-{NL / xxxxxx},TYPE-CLEAR,ASCII-{YES / NO},
	DENSITY-{1600 / 800 / 556 / 200},ID-yy...yy
	// END

Display volume label.	// VOL UNIT-{T1 / T2 / T3 / T4},TYPE-DISPLAY,DENSITY-{800 / 556 / 200} // END

Notes:

1. If density is not specified, the default for 7-track tape units is 800 bpi, the default for 9-track tape units is 1600 bpi.
2. The DENSITY parameter on display volume label is valid only for 7-track tape units.
3. Valid density for 7-track tape units is 200, 556, and 800 bpi. Valid density for 9-track tape units is 800 bpi (if dual density feature is installed) and 1,600 bpi.

PARAMETER SUMMARY

TYPE-CHECK	Checks to see if the file has expired, then writes a new label. Do not use this on blank tapes because the program attempts to read a blank tape, causing tape runaway.
TYPE-CLEAR	Writes a new volume label without checking for an expired file.
TYPE-DISPLAY	Prints the contents of the volume label and the header labels.
UNIT-code	Specifies which tape drive contains the tape to be initialized. Possible codes are T1, T2, T3, and T4. A separate VOL statement is needed for each tape unit that contains a tape to be initialized.
REEL-NL	Specifies that an unlabeled tape is to be generated.
REEL-xxxxxx	Specifies the volume serial number that the Tape Initialization program writes on tape. Must be alphabetic A-Z, @, #, $, or numeric 0-9.
ASCII-YES	The tape is written in ASCII code. This is invalid for 7-track tape.
ASCII-NO	The tape is written in EBCDIC code. If the ASCII parameter is omitted, NO is assumed.
DENSITY-200	The tape is written at a density of 200 bpi. The file written on this tape unit must be written at this density.
DENSITY-556	The tape is written at a density of 556 bpi. The file written on this tape unit must be written at this density.
DENSITY-800	The tape is written at a density of 800 bpi. The file written on this tape unit must be written at this density.
DENSITY-1600	The tape is written at a density of 1,600 bpi. The file written on this tape unit must be written at this density.
ID-xxxxxxxxxx	Provides an additional identification field. This field is not processed by the system. A maximum of 10 characters can be used if ASCII-NO is specified. If ASCII-YES is specified, 14 characters can be used. This is an optional parameter.

OCL CONSIDERATIONS

The following OCL statements are needed to load the Tape Initialization program:

```
// LOAD $TINIT,code
// RUN
```

The code you supply depends on the location of the simulation area containing the Tape Initialization program. The codes are R1, F1, R2, and F2.

MESSAGE FOR TAPE INITIALIZATION

Message	Meaning
INITIALIZATION ON xx COMPLETE	This message is printed when initialization of a tape is complete. xx indicates the unit (T1, T2, T3, or T4) on which the initialization is complete.

PRINTOUT OF VOLUME LABEL

The following sample jobs show the format of data printed by the Tape Initialization program from a 9-track tape unit and from a 7-track tape unit.

```
// LOAD $TINIT,F1
// RUN
// VOL UNIT-T1,TYPE-DISPLAY,FILES-ALL,DENSITY-200
// VOL UNIT-T2,TYPE-DISPLAY,FILES-ALL,DENSITY-556
// VOL UNIT-T3,TYPE-DISPLAY,FILES-ALL
// VOL UNIT-T4,TYPE-DISPLAY,FILES-ALL
// END
                                  ***  DISPLAY ON UNIT T1  ***
 LABEL   SERIAL   OWNER CODE
 VOL1    SCRT01
-------------------------------------------------------------------------------------------------------------
 LABEL     FILE IDENTIFIER        FILE SERIAL      VOL SEQ NO   CREATE DATE   EXPIRE DATE   FILE SEQ NO
 HDR1      FILE01                   SCRT01             0001         76006        76016          0001
 LABEL   REC FORM   BLK LENG   REC LENG   RECORDING TECH   PRTR CNTRL   BLK ATTR     JOBNAME/STEPNAME
 HDR2       F        00080      00080                                                    /TAPBLD01
-------------------------------------------------------------------------------------------------------------

                                  ***  DISPLAY ON UNIT T2  ***
 LABEL   SERIAL   OWNER CODE
 VOL1    SCRT02
-------------------------------------------------------------------------------------------------------------
 LABEL     FILE IDENTIFIER        FILE SERIAL      VOL SEQ NO   CREATE DATE   EXPIRE DATE   FILE SEQ NO
 HDR1      FILE01                   SCRT02             0001         76006        76016          0001
 LABEL   REC FORM   BLK LENG   REC LENG   RECORDING TECH   PRTR CNTRL   BLK ATTR     JOBNAME/STEPNAME
 HDR2       F        00800      00080                                      B                    /TAPBLD01
-------------------------------------------------------------------------------------------------------------

                                  ***  DISPLAY ON UNIT T3  ***
 LABEL   SERIAL   OWNER CODE
 VOL1    SCRT03
```

MEANING OF VOLUME LABEL INFORMATION

Display of Volume Label

Heading	Meaning
LABEL	VOL1 indicates this is a volume label.
SERIAL	The volume serial number (from the REEL parameter).
OWNER CODE	Additional identification (from the ID parameter).

Display of Header 1 Label

Heading	Meaning
LABEL	HDR1 indicates this is a header 1 label.
FILE IDENTIFIER	The filename of the file on tape. This is the name from the LABEL parameter of the OCL FILE statement when the file was created.
FILE SERIAL	The serial number of the tape volume. This is the same as the SERIAL field in the volume label.
VOL SEQ NO	The sequence number of this volume is a multivolume file.
CREATE DATE	The date this file was created. This is a Julian date. The format is yyddd where yy is the last two digits of the year and ddd is the day in the year. Example: 76063 = the sixty-third day of 1976, or March 3, 1976.
EXPIRE DATE	The date this file expires. This Julian date is the creation date plus the number of days specified by the RETAIN parameter on the OCL FILE statement.

Display of Header 2 Label

Heading	Meaning
LABEL	HDR2 indicates this is a header 2 label.
REC FORM	The record format of this file. (From the RECFM parameter on the OCL FILE statement when this file was created.) The formats are: F Fixed length V Variable length U Undefined length
BLK LENG	Block length (from the BLKL parameter on the OCL FILE statement when this file was created).
REC LENG	Record length (from the RECL parameter on the OCL FILE statement when this file was created).
RECORDING TECH	T Odd parity with translation C Odd parity with conversion E Even parity without translation ET Even parity with translation Blank Odd parity without translation or conversion
PRTR CNTRL	Printer control character. This field will be blank on tapes created on System/3. For tapes created on other systems, the characters are: A ASCII control characters M Machine control characters blank No control characters
BLK ATTR	Block attributes: B Blocked records S Spanned records R Blocked and spanned records blank Neither blocked nor spanned *Note:* Spanned records cannot be created on System/3.

Tape Error Summary Program—$TVES

The IBM System/3 Disk System keeps track of errors that occur on the tape drives. This error information is stored in the customer engineer tracks on drive 1. You should run the Tape Error Summary program periodically to provide a summary, by volume and by unit, of temporary read and write errors.

There are no control statements necessary for this program. After being loaded from the program or system pack, the Tape Error Summary program reads the data from the disk and sorts it by volume and unit. When all the data is read or the available main storage is filled, the error data is printed. If no tape errors are recorded, the message THERE ARE NO VALID TAPE ERRORS LOGGED is printed.

ERROR LOGGING FORMAT

```
SUMMARY MAGNETIC TAPE ERROR STATISTICS BY VOLUME  DATE 03/27/72
  (1)        (2)       (3)       (4)       (5)
VOLUME     SIO       TEMP      TEMP      WRITE
SERIAL     COUNT     READ      WRITE     SKIP

T1         06512     0000      0028      0028
TAPE1      00016     0000      0001      0001
TAPE3      00021     0000      0001      0001

SUMMARY MAGNETIC TAPE ERROR STATISTICS BY TAPE UNIT  DATE 03/27/72
            (2)       (3)       (4)       (5)       (6)
TAPE       SIO       TEMP      TEMP      WRITE     DIAG
UNIT       COUNT     READ      WRITE     SKIP      TRACK

T1         06528     0000      0029      0029      0000
T4         00021     0000      0001      0001      0000
```

(1) For any file that has more than two volumes on a unit, ,,,,,, is printed as the volume serial for all volumes on that unit except the last volume. For a tape that is not being used by tape data management, ,,,,,, is printed as the volume serial. For nonlabeled tapes, ****** is printed as the volume serial. For tapes with nonstandard labels, NS is printed as the volume serial.

(2) The number of tape operations performed. (SIO means Start I/O.)

(3) Temporary read errors.

(4) Temporary write errors.

(5) Write skips caused by temporary write errors.

(6) Diagnostic track errors. This is used by IBM customer engineers.

OCL CONSIDERATIONS

The following OCL statements are needed to load the Tape Error Summary program:

```
// LOAD $TVES,code
// RUN
```

The code you supply depends on the location of the simulation area containing the Tape Error Summary program. The codes are R1, F1, R2, and F2.

Disk Initialization Program—$INIT

All data modules must be initialized before use. Data modules that have been initialized need not be reinitialized unless you want to erase their contents and rename them.

The Disk Initialization program prepares data modules for use. It does this by:

- Writing track and record addresses on the data module
- Checking for defective tracks, a process called surface analysis
- Assigning alternate tracks to any defective tracks found
- Writing a name on each data module to identify the data module
- Formatting the volume table of contents

The process is called initialization. The program can initialize up to two data modules during the same program run.

There are five types of initialization: FORCE, PRIMARY, CLEAR, CYL0, and RENAME. FORCE is used primarily to initialize new data modules. PRIMARY is used to initialize the main data area if there are no active files on the data module. CLEAR will initialize the main data area without checking for active files. CYL0 is a fast initialization, initializing only cylinder 0 on a System/3 formatted data module. RENAME affects names on cylinder 0, track 0, record 3 and cylinder 0, track 3, record 3.

CAUTION
CLEAR and FORCE destroy any files that were previously on disk. CYL0 destroys any VTOC entries that were previously on disk.

The control statements you supply for the Disk Initialization program depend on the type of initialization and the number of disks you are initializing.

CONTROL STATEMENT SUMMARY

Type of Initialization		Control Statements ①
FORCE ②		// UIN TYPE-FORCE ③,UNIT-D2
		// VOL PACK-name,ID-characters,NAME360-characters
		// END
PRIMARY ②		// UIN TYPE-PRIMARY ③,UNIT-{code / codes},VERIFY-number
		// VOL PACK-name,ID-characters,NAME360-characters
		// END
Disk already in use (reinitialize)		// UIN TYPE-PRIMARY,UNIT-{code / codes},VERIFY-number
		// VOL PACK-name,ID-characters,NAME360-characters,OLDPACK-name
		// END
CLEAR ②	Disk already in use	// UIN TYPE-CLEAR,UNIT-{code / codes},VERIFY-number
		// VOL PACK-name,ID-characters,NAME360-characters,OLDPACK-name
		// END
CYL0 ②		// UIN TYPE-CYL0,UNIT-{code / codes},
		// VOL PACK-name,ID-characters,NAME360-characters,OLDPACK-characters
		// END
RENAME ②		// UIN TYPE-RENAME,UNIT-{code / codes},
		// VOL PACK-name,ID-characters,NAME360-characters,OLDPACK-characters
		// END

Note: The control statement defaults to TYPE-FORCE if the data module is still in System/370 format and TYPE-CLEAR or PRIMARY initialization has been specified. If CYL0 or RENAME is specified and the data module is still in System/370 format, the system halts.

---

① Control statements are required in the order they are listed: UIN, VOL, END

② One VOL statement is required for each disk listed in the UNIT parameter of the UIN statement. The PACK parameter in the first VOL statement applies to the first disk listed in the UNIT parameter. The PACK parameter in the second VOL statement applies to the second disk listed in the UNIT parameter.

③ If the TYPE parameter is omitted, TYPE-PRIMARY is assumed.

PARAMETER SUMMARY

UIN (Input Definition) Statement

TYPE-FORCE	If the TYPE parameter FORCE is used, the main data area and the simulation areas are initialized without a check for active files. (This is invalid for D1; and for D2 if F2 and R2 are being simulated.)
TYPE-PRIMARY	Primary initialization (main data area only). Tracks already initialized are reinitialized. The program will not initialize disks containing temporary data files or permanent data files.
TYPE-CLEAR	Clear initialization (main data area only). Tracks already initialized are reinitialized. Active file checking is bypassed and any data on the tracks is destroyed.
TYPE-CYL0	CYL0 is a fast initialization, initializing only cylinder 0 on a System/3 formatted data module. This includes rewriting the volume label, the pack ID, and NAME360 fields, and deleting any VTOC entries that may be present.
TYPE-RENAME	RENAME initialization applies only to those names on cylinder 0 which match the PACK, ID, and NAME360 parameters. Parameters are changed on a System/3 formatted data module to the parameters specified on the control statement.
UNIT-code	Disk location (one disk). Possible codes: D1, D2.
UNIT-'code,code'	Disk location (two disks). Possible codes: D1, D2.
VERIFY-number	Surface analysis. Done the number of times indicated (number can be 1-255). VERIFY-16 is assumed if you omit the parameter. This parameter is only used for TYPE-CLEAR and TYPE-PRIMARY initialization.
ERASE-code	Possible codes are yes and no. Yes causes retesting of tracks for which alternates are already assigned.

VOL (Volume) Statement

PACK-name	Data module name. Can contain any of the standard System/3 characters except apostrophes, leading or embedded blanks, and embedded commas①. Its length must not exceed 6 characters.
ID-characters	Additional identification. Can contain any of the standard System/3 characters except apostrophes, leading or embedded blanks, and embedded commas①. Its length must not exceed 10 characters. If you omit this parameter, no additional identification is written on the disk.
NAME360-characters	Additional identification for data module. The name will be placed in the System/360 format 1 DSCB. Can contain any of the standard System/3 characters except apostrophes, leading or embedded blanks, and embedded commas①. Its length must not exceed 44 characters. If you omit this parameter, the program defaults to SYSTEM/3.DATA.
OLDPACK-name	Current name of the data module to be initialized. See PACK keyword (above) for valid entries.

PARAMETER DESCRIPTIONS

TYPE Parameter (UIN)

The TYPE parameter indicates the type of initialization you want to do: PRIMARY, FORCE, CLEAR, CYL0, or RENAME. The type of initialization determines which disk tracks will be initialized.

PRIMARY Initialization

PRIMARY initialization applies to main data areas you have used but want to initialize again. Tracks that were previously initialized are initialized again. Any data on the tracks is destroyed. You can use PRIMARY initialization on a disk as often as you want. However, the program will not initialize disks containing temporary data files or permanent data files. You must delete the files using the file delete program.

① This is due to their delimiter function.

FORCE Initialization

FORCE initialization applies to new data modules that are formatted for System/370. FORCE may also be used to reinitialize disks that you have used.

Note: The simulation area program, $SCOPY must be used after a FORCE initialization to reformat the simulation areas.

CLEAR Initialization

CLEAR initialization applies to the main data area of previously used data modules that require reinitialization because of invalid data module labels or an unrecoverable disk error. Tracks that were previously initialized are reinitialized.

CAUTION
All temporary data files or permanent data files are completely erased.

CYL0 Initialization

Cylinder zero (CYL0) initialization can be used if you want to reinitialize only cylinder 0.

RENAME Initialization

RENAME initialization may be used if you want to change PACK, ID, and NAME360 parameters.

Note: If an invalid System/3 label is found during RENAME initialization, the program must reinitialize the disk using either FORCE, CLEAR, PRIMARY, or CYL0.

UNIT Parameter (UIN)

The UNIT parameter (UNIT-code) indicates the location of the data modules you want to initialize. The program can initialize up to two data modules during one program run.

The form of the UNIT parameter depends on the number of data modules you are initializing:

- For one data module, use UNIT-code.
- For two data modules, use UNIT-'code,code'.

The codes indicate the locations of the data modules D1, D2.

For all initialization, the order of codes must correspond to the order of VOL control statements. If, for example, you had used the parameter UNIT-'D1,D2', the first VOL statement applies to the data module on drive 1, and the second to the data module on drive 2.

ERASE Parameter (UIN)

The ERASE parameter applies to alternate track assignments on disks that have already been intialized and used, but you are reinitializing using primary initialization.

The condition of tracks on disks such as these has been tested at least once before (during the previous initialization), and the tracks that were found to be defective during surface analysis were assigned alternates. The ERASE parameter allows you to indicate whether you want the program to 1) retest the tracks to which alternate tracks are already assigned, or 2) leave the alternate tracks assigned without retesting the tracks.

To retest the tracks, enter parameter ERASE-YES. The program then erases any existing alternate track assignments and tests all tracks as if the disk were new.

To bypass retesting the tracks, enter parameter ERASE-NO. The program then tests only those tracks to which no alternate tracks are assigned. Alternate tracks previously assigned remain assigned.

Defective tracks are not retested if the ERASE parameter is omitted.

VERIFY Parameter (UIN)

The VERIFY parameter (VERIFY-number) concerns surface analysis. It enables you to indicate the number of times you want the program to do surface analysis before judging whether or not tracks are defective. The number can be from 1 to 255.

Surface Analysis

Surface analysis is a procedure for testing the condition of tracks. It consists of writing test data on tracks, then reading the data to ensure that it was recorded properly.

In judging whether or not tracks are defective, the program does surface analysis the number of times you specify in the VERIFY parameter. If you omit the VERIFY parameter, surface analysis is done 16 times. Tracks that cause reading or writing errors any time during surface analysis are considered defective. Defective tracks can be assigned alternates. The 3340 has 40 alternate tracks available. If the program finds more than 40 defective tracks, it considers the disk unusable and stops initializing it.

Alternate Track Assignment

Alternate track assignment is the process of assigning an alternate track to a defective track. If the disk initialization program finds a defective track during surface analysis, it assigns an alternate track to the defective track. The alternate is, in effect, a substitute for the defective track. Anytime a program attempts to use the defective track, it automatically uses the alternate instead. Each disk has 40 alternate tracks (tracks 3340-3379).

If tracks become defective after a disk is initialized, another program (see *Alternate Track Assignment Program*) is used to assign alternate tracks. Disks need not be reinitialized to assign alternate tracks.

Note: During initialization of D1, suspected defective simulation area tracks may be encountered in the suspected defective track list as a result of previous activity involving that data module. If so, the system halts at end of job. At this time the data module should be moved to unit D2 and $ALT run against it.

PACK Parameter (VOL)

The PACK parameter (PACK-name) applies to all types of initialization. During initialization, the disk initialization program writes a name on each disk. It uses the name you supply in the corresponding PACK parameter. (One VOL control statement containing a PACK parameter is required for each disk.)

The name can be any combination of standard System/3 characters except apostrophes, leading or embedded blanks, and embedded commas (because of their delimiter function). (See Appendix A for a list of standard System/3 characters.) Its length must not exceed 6 characters. Examples of valid disk names are 0,F0001, 012, A1B9, and ABC.

In general, disk names are used for checking. Before a program uses a disk, the disk name is compared with a name you supply (either in OCL statements or control statements required by the program). If the names do not match, the program halts and prints a message. In this way, programs cannot use the wrong disks without the operator knowing about it.

ID (Identification) Parameter (VOL)

The ID parameter (ID-characters) applies to all types of initialization. It enables you to include a maximum of 10 characters, in addition to the disk name, to further identify a disk. The characters can be any combination of standard System/3 characters (Appendix A) except apostrophes, leading or embedded blanks, and embedded commas (because of their delimiter function). The information is strictly for your use; the system does not use it for checking. If you use the file and volume label display program to print the disk name, that program will also print the additional identification for you.

NAME360 Parameter (VOL)

The NAME360 parameter (NAME360-name) is used to specify a filename for data interchange with System/360-System/370. System/360-System/370 can use data on a System/3 data module by treating it like a file. System/3 gives a default filename of SYSTEM/3.DATA. The NAME360 parameter can be used if you would like to code a filename of your own.

NAME360 can contain any of the standard System/3 characters except apostrophes, blanks and commas. Its length must not exceed 44 characters.

OLDPACK Parameter (VOL)

The OLDPACK parameter (OLDPACK-name) is used to verify that a specific disk is mounted before initialization is started. If the name of the disk mounted does not match the name you specify, the program halts.

The specified name can be any combination of standard System/3 characters except apostrophes, leading or embedded blanks, and embedded commas. Its length must not exceed 6 characters.

OCL CONSIDERATIONS

The following OCL statements are needed to load the disk initialization program:

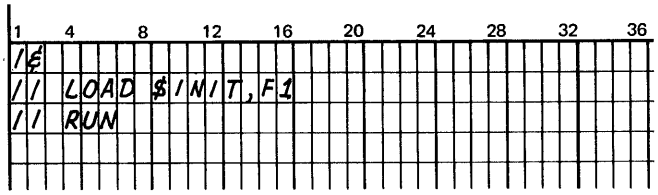
```
// LOAD $INIT,code

// RUN
```

The code you supply depends on the location of the simulation area containing the disk initialization program. The codes are R1, F1, R2, and F2.

EXAMPLES

Primary Initialization of Two Disks

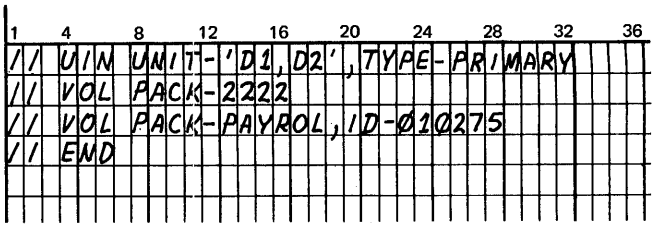
Figures 9 and 10 are examples of OCL statements and utility control statements needed for the primary initialization of two disks.



Explanation:

Disk initialization program is loaded from the simulation area (F1) on drive 1.

Figure 9. OCL Load Sequence for Disk Initialization



Explanation:

- The main data area on both drives is being initialized (UNIT-'D1,D2' in UIN statement).
- The main data area (D1) is given the name 2222 (PACK-2222 in first VOL statement).
- The main data area (D2) is given the name PAYROL (PACK-PAYROL in second VOL statement). Additional identifying information, 010275, is to be written on drive 2 (ID-010275).

Figure 10. Utility Control Statements for Primary Initialization of Two Disks

MESSAGES FOR DISK INITIALIZATION

Message	Meaning
INITIALIZATION ON XX COMPLETE	This message is printed when initialization of a disk is complete. XX indicates the unit (D1,D2) on which the initialization is complete.
INITIALIZATION ON XX TERMINATED	This message is printed when initialization of a disk must be terminated for one of the following reasons: • Cylinder 0 head 0 is defective. • More than forty 3340 tracks are defective. • Possible disk hardware error exists. After this message is printed, halt 33 occurs. XX indicates the unit (D1 or D2) on which the initialization is terminated.
ALTERNATE TRACKS ASSIGNED PRIMARY TRACK XXXX ALTERNATE TRACK XXXX	These two messages are printed when a primary track is defective and an alternate track is assigned to it. XXXX indicates the tracks involved.
ALTERNATE TRACK XXXX DEFECTIVE	This message is printed when a 3340 alternate track is defective.
PRIMARY TRACK HAS BEEN TESTED OK TRACK-XXXX, UNIT-ZZ	This message is printed when it is determined that a primary track is not defective. XXXX is the primary track number and ZZ is the unit involved.
RECORD WITH DATA ERROR	This message is printed when an error is encountered during data transfer while assigning an alternate track. The record that has the error is printed. (See *Alternate Track Assignment Program* for additional explanation.)

Alternate Track Assignment Program—$ALT

The alternate track assignment program assigns alternate tracks to disk tracks that become defective after they are initialized. When the program assigns an alternate, it transfers the contents of the defective track to the alternate. Alternate tracks can replace any primary tracks except cylinder 0 head 0 on the 3340 because they are reserved for system use.

CONTROL STATEMENT SUMMARY

Use	Control Statements (1)
Conditional Assignment	// ALT (2) PACK-name,UNIT-code,VERIFY-number
	// END

(1) For each use, the program requires the statements in the order they are listed: ALT, END.

(2) There can be only two ALT statements per job.

PARAMETER SUMMARY: ALT (ALTERNATE) STATEMENT

PACK-name	Name of the disk.
UNIT-code	Location of the disk. Possible codes are D1, D2.
VERIFY-number	In testing the condition of a track, do surface analysis the number of times indicated (number can be 1-255). If VERIFY parameter is omitted, do surface analysis 16 times.

PARAMETER DESCRIPTIONS

PACK Parameter

The PACK parameter (PACK-name) tells the program the name of the disk containing the defective tracks. This is the name written on the disk by the disk initialization program. (See *Disk Initialization Program.*)

The alternate track assignment program compares the name in the PACK parameter with the name on the disk to ensure they match. In this way, the program ensures that it is using the right disk.

UNIT Parameter

The UNIT parameter (UNIT-code) indicates the location of the disk containing defective tracks. Codes for the possible locations are D1 and D2.

VERIFY Parameter

The VERIFY parameter (VERIFY-number) enables you to indicate the number of times you want the program to do surface analysis before judging whether or not the track is defective. The number can be from 1 to 255. If you omit the parameter, the program does surface analysis 16 times.

Conditional Assignment

Conditional assignment consists of testing the condition of a track (surface analysis) and, if the track is defective, assigning an alternate track to replace it.

Situation: Conditional assignment applies to tracks that cause reading or writing errors during a job. Any time a track causes such errors, the system does the following:

1. Stops the program currently in operation.
2. Writes the track address in a special area on the disk.
3. Halts with a halt code indicating a permanent disk I/O error.

   You can then run the alternate track assignment program.

When you use the alternate track assignment program to do conditional assignment, the program locates the tracks by using the addresses in the special area on disk. All disks have such an area. The program will do conditional assignment for all tracks identified in the area (one at a time) as long as there are alternate tracks available for assignment.

Surface Analysis: Surface analysis is a procedure the program uses to test the condition of tracks. It consists of writing test data on a track, then reading the data to ensure that it was written properly.

Before doing surface analysis, the alternate track assignment program transfers any data from the track to an alternate track. This is the alternate that will be assigned if the track proves to be defective.

In judging whether or not the track is defective, the program does surface analysis the number of times you specify in the VERIFY parameter. If you omit the parameter, the program does surface analysis 16 times. If the track causes reading or writing errors any time during surface analysis, the program considers the track defective.

Assignment of Alternate Tracks: If a track proves to be defective, the program assigns an alternate track. The alternate becomes, in effect, a substitute for the defective track. Any time a program attempts to use the defective track, it automatically uses the alternate instead.

Each data module has 40 alternate tracks. The program will not do conditional assignment if all alternate tracks are in use.

Note: If the alternate track assignment program is being run against D1 and suspected defective tracks from the simulation area are encountered, a halt occurs at end of job. At this time, the data module should be moved to D2 and $ALT run against it again.

Incorrect Data: If a track is defective, some of the data transferred to the alternate track could be incorrect. Therefore, when reading data from the defective track, the program prints all track records containing data that caused reading errors. Characters that have no print symbol are printed as two-digit hexadecimal numbers.

The following is an example:

```
ABCDE GH123 56...
    B       A
    6       4
```

Appendix A lists the characters in the standard character set and their corresponding hexadecimal numbers.

To correct errors on the alternate track, use the alternate track rebuild program.

OCL CONSIDERATIONS

The following OCL statements are needed to load the alternate track assignment program:

```
// LOAD $ALT,code
// RUN
```

The code you supply depends on the location of the simulation area containing the alternate track assignment program. The codes are as follows: R1, F1, R2, and F2.

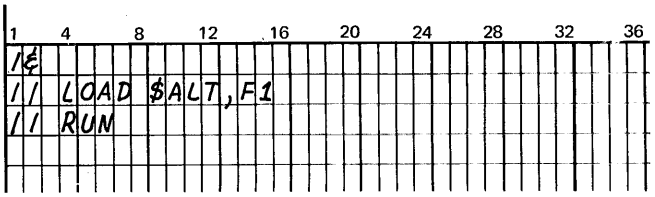
EXAMPLES

Conditional Assignment

Figures 11 and 12 are examples of the OCL statements and utility control statements needed for a conditional assignment as described in the following situation.

Situation

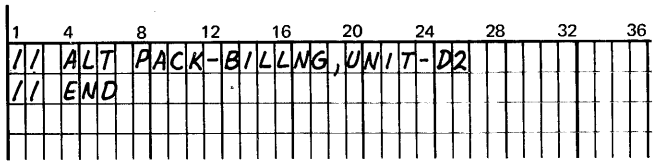
The system cancels a job if a defective track is found on the main data area on drive 2. (The name of the disk is BILLNG.) Before doing more jobs, the operator wants to use the alternate track assignment program to check the condition of the track and assign an alternate to the track if it is defective.



Explanation:

Alternate track assignment program is loaded from the simulation area F1 on drive 2.

Figure 11. OCL Load Sequence for Alternate Track Assignment



Explanation:

- The name of the disk (BILLNG) and its location (main data area on drive 2) are indicated by the PACK and UNIT parameters in the ALT statement.
- Because we omitted the VERIFY parameter from the ALT statement, the program does surface analysis 16 times when it tests the condition of the tracks.

Figure 12. Utility Control Statements for a Conditional Assignment

MESSAGES FOR ALTERNATE TRACK ASSIGNMENT

Message	Meaning
ALTERNATE TRACK ASSIGNED	This message is printed when an alternate track has been assigned to a defective track and the data has been transferred to the alternate track.
PRIMARY TRACK HAS BEEN TESTED OK TRACK xxxx,UNIT-zz	This message is printed when it is determined that a primary track is not defective. xxxx is the primary track number and zz is the unit involved.
RECORD WITH DATA ERROR	This message is printed when the alternate track assignment program found an error when transferring data. The record that has the error is printed out.
PRIMARY TRACK xxxx ALTERNATE TRACK yyyy,UNIT-zz	This message is printed after ALTERNATE TRACK ASSIGNED. xxxx is the primary track number, yyyy is the alternate track number, and zz is the unit involved.

Alternate Track Rebuild Program—$BUILD

The alternate track rebuild program enables you to correct data that could not be transferred correctly to an alternate track. One or more alternate tracks can be corrected during a program run. You must supply the control statements and data used to correct the errors.

In writing control statements for this program, you will need the information printed by the alternate track assignment program when it assigned the alternate track. The printed information tells you the name of the disk and numbers of the track and records suspected of containing incorrect data. It also includes the data from these records, which you can use to locate incorrect data. On the 3340, fixed record refers to a physical 256-byte record.

CONTROL STATEMENT SUMMARY ①

```
// REBUILD PACK-name,UNIT-code,TRACK-location,
LENGTH-number,DISP-position

Substitute data

// END
```

---

① To replace characters 1-12 and 75-78 of a record, you can use either of the following:

- Use one REBUILD statement to replace all the characters with a LENGTH parameter of 78.
- Use one REBUILD statement for every set of positions you correct.

The data you want to substitute must follow the REBUILD statements to which it applies. The order of the statements and data in the preceding example would be:

```
// REBUILD statement data      For positions 1-78
// END

// REBUILD statement data      For positions 1-12
// REBUILD statement data      For positions 75-78
// END
```

PARAMETER AND SUBSTITUTE DATA SUMMARY

REBUILD Statement

PACK-name	Name of the disk.
UNIT-code	Location of the disk. Possible codes are D1 and D2.
TRACK-location	*3340 Disk Unit*—Number of track and fixed record containing incorrect data. Number is printed by alternate track assignment program. Track number must be four digits; fixed record number must be two digits. (TRACK-011109 means track 111, fixed record 9.)
LENGTH-number	Number of characters being replaced. Number can be 2-256 and must be a multiple of 2 (2, 4, 6, etc).
DISP-position	Position of the first character being replaced in the record. Position can be 1-255.

Substitute Data: Code each character in hexadecimal form. Follow every second character, except the last, with a comma. Example: The numbers 123456 would be coded as F1F2,F3F4,F5F6. (Appendix A lists the hexadecimal codes for System/3 characters.)

PARAMETER AND SUBSTITUTE DATA DESCRIPTIONS

PACK Parameter

The PACK parameter (PACK-name) tells the program the name of the disk that contains the alternate track being corrected. This name is the one written on the disk by the disk initialization program.

The alternate track rebuild program compares the name in the PACK parameter with the name on the disk to see if they match. In this way, the program ensures that the program is using the right disk.

UNIT Parameter

The UNIT parameter (UNIT-code) indicates the location of the disk that contains the alternate track being corrected. Codes for the possible locations are D1 and D2.

TRACK Parameter

The TRACK parameter (TRACK-location) identifies the track and record containing the data being corrected. The defective track, not the alternate track, is the one you refer to. Referencing the defective track is the same as referencing the alternate track.

For the main data area, the possible track numbers are 0001-4184. Always use four digits. The possible fixed record numbers are 01-48. Always use two digits. The track number must precede the fixed record number. For example, the parameter TRACK-111019 means track 1110, record 19.

Track and record numbers are printed by the alternate track assignment program when it prints data from records that contain incorrect data.

LENGTH Parameter

The LENGTH parameter (LENGTH-number) tells the program how many characters you are replacing in the fixed record. You must replace characters in multiples of 2 (2, 4, 6, and so on). The maximum is 256, which is the capacity of a fixed record.

Length applies to characters that occupy consecutive positions in the fixed record. If the characters you want to replace do not occupy consecutive positions, you must either replace all intervening characters or use more than one REBUILD statement. For example, to replace characters 10-11 and 24-25 in a fixed record, you can do either of the following:

- Use one REBUILD statement to replace characters 10-25 (LENGTH-16).
- Use two REBUILD statements to replace characters 10-11 (LENGTH-2) and 24-25 (LENGTH-2).

DISP (Displacement) Parameter

The DISP parameter (DISP-position) indicates the position of the first character being replaced in the fixed record. The position of the first character is 1; the position of the second character is 2, and so on. The maximum position you can specify is 255.

Beginning at the position you indicate, the alternate track rebuild program replaces the number of characters you indicate in the LENGTH parameter.

Substitute Data

After each REBUILD statement, you must code the substitute characters that apply to that statement. The characters must be in hexadecimal form. Appendix A shows the hexadecimal codes for the System/3 character set.

Include a comma after every second character. For example, the data F1F2,F3F4,F5F6 represents 123456. F1 is the hexadecimal form of 1; F2 is the hexadecimal form of 2, and so on.

Code only the number of characters you indicated in the LENGTH parameter in the REBUILD statement.

Note: If the LENGTH parameter of the REBUILD statement exceeds 38, at least two substitute data statements are required. Each substitute data statement, except the last one, must be completely filled with data and must have a comma in column 95 and a blank in column 96. If the 1442 is the only input device, it is possible to have only one substitute data statement.

OCL CONSIDERATIONS

The following OCL statements are needed to load the alternate track rebuild program.

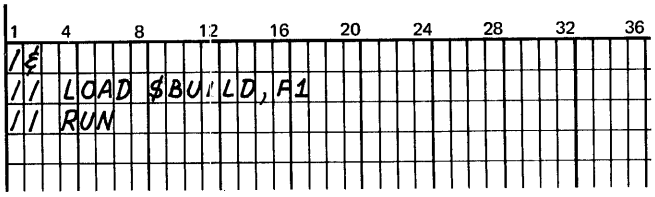
```
// LOAD $BUILD, code
// RUN
```

The code you supply depends on the location of the simulation area contining the alternate track rebuild program. The codes are R1, F1, R2, and F2.

EXAMPLES

Correcting Characters on an Alternate Track

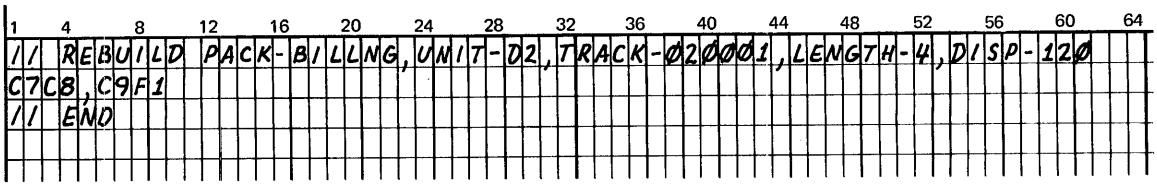
Figures 13 and 14 are examples of the OCL and utility control statements needed for correcting characters on an alternate track.



Explanation:

Alternate track rebuild program is loaded from the simulation area F1 on drive 1.

Figure 13. OCL Load Sequence for Alternate Track Rebuild



Explanation:

- The name of the main data area (BILLNG) and its location (drive 2) are indicated in the PACK and UNIT parameters in the REBUILD statement.
- The record containing the incorrect characters is record 1 of the alternate track assigned to track 200 (TRACK-020001). The character in position 120 is the first character being replaced (DISP-120).
- The characters in positions 120 through 123 in record 1 are being replaced (LENGTH-4).
- The substitute characters follow the REBUILD statement. They are G (C7), H (C8), I (C9), and 1 (F1).

Figure 14. Utility Control Statements for Correcting Characters on an Alternate Track

Situation

Assume that the alternate track assignment program printed the following information:

```
**RECORD WITH DATA ERROR**
TRACK    1.......10.......20.......30.......40.......50.......60.......70.......80......88
002001
         00000000000000001111111111111111222222222222222233333333333333333 44444444444444455555555
         01234567890ABCDEF01234567890ABCDEF01234567890ABCDEF01234567890ABCDEF 12345678ABCDEF01234567

         55555555666666666666666777777777777777788888888888888899999999999999AAAAAAAAAAAAAAAA
         89ABCDEF01234567890ABCDEF01234567890ABCDEF01234567890ABCDEF01234567890ABCDEF01234567890ABCDEF
                        ABCDEFGHI        JKLMNOPQR        STUVWXYZ       0123456789
         BBBBBBBBBBBBBBBC          CCCCCCD          DDDDDDEF         EEEEEE           FFFFFF
         01234567890ABCDEFO        ABCDEFO          ABCDEF01         ABCDEF           ABCDEF
```

It means that errors were detected in record 1 of track 200. (Assume the name of the disk is BILLNG.)

In checking the characters printed by the program, you found that the characters in positions 120-123 in the record are incorrect and you want the operator to run the alternate track rebuild program to correct them.

File and Volume Label Display Program—$LABEL

The file and volume label display program has two uses:

- Print the entire volume table of contents (VTOC) from a disk.
- Print only the VTOC information for certain data files.

In both cases, the program also prints the name of the disk.

The printed VTOC information is a readable, up-to-date record of the contents of the disk. There can be any number of reasons why you might need the information. Some of the more common ones are as follows:

- Before reinitializing a disk, you might want to check its contents to ensure that it contains no libraries, permanent data files, or temporary data files.
- You want to find out what disk areas are available for libraries or new files.
- You want specific file information, such as the file name, designation (permanent, temporary, scratch), or the space reserved for the file.

The control statements you supply for the program depend on the program use.

CONTROL STATEMENT SUMMARY

Uses	Control Statement①
Print entire VTOC	// DISPLAY UNIT-code,LABEL-VTOC,SORT-NAME,FORMAT-{A/B}
	// END
Print only file information from VTOC	// DISPLAY UNIT-code,LABEL-{filename / filenames}②,FORMAT-{A/B}
	// END

① For each use, the program requires the statements in the order they are listed: DISPLAY, END.

② The number of filenames you list for a program run may not exceed 20. (VTOC is considered as one filename.)

PARAMETER SUMMARY (DISPLAY STATEMENT)

UNIT-code	Location of the disk containing the VTOC information being printed. Possible codes are R1, F1, R2, F2, D1, and D2.
LABEL-VTOC	Print entire contents of VTOC.
LABEL-filename	Print VTOC information for one file.
LABEL-'filename,filename,...'	Print VTOC information for more than one file.①
SORT-NAME	VTOC information is sorted by filename into alphabetical order.
FORMAT-A	To be used when 120 print positions are available.
FORMAT-B	To be used when 96 print positions are available; prints two lines for each VTOC entry.

①The number of filenames you list for a program run may not exceed 20. (VTOC is considered as one filename.)

PARAMETER DESCRIPTIONS

UNIT Parameter

The UNIT parameter (UNIT-code) indicates the location of the disk containing the VTOC information being printed. Codes for the possible locations are R1, F1, R2, F2, D1, and D2.

LABEL Parameter

The LABEL parameter indicates the information you want printed: the entire contents of the VTOC or only the information for certain files. The VTOC is an area on disk that contains information about the contents of the disk.

SORT Parameter

The SORT parameter can be specified only when LABEL-VTOC is specified. If SORT-NAME is specified, the VTOC information is sorted by filename into alphabetical order. This function applies only to 1,000-file VTOCs and requires additional main storage for sorting, as shown below:

Number of VTOC Entries	Storage Required for Execution With Sort
1-300	10K
301-500	12K
501-700	14K
701-1000	16K

FORMAT Parameter

If the system you are using has at least 120 print positions, FORMAT-A is the default and only acceptable option. If the system has a printer with 96 print positions, FORMAT-A truncates the print line to omit NEXT AVAIL REC and NEXT AVAIL KEY. FORMAT-B causes the NEXT AVAIL REC and NEXT AVAIL KEY to be printed on the next line.

ENTIRE CONTENTS OF VTOC

The parameter LABEL-VTOC means to print the entire contents of the VTOC. The meaning of the information the program prints is given in the following chart. Headings that are listed are the ones printed by the program to identify the information. Figures 15 and 16 are examples of VTOC printouts.

If the program needs more than one page to list the file information, it prints the headings for the file information at the top of each new page.

```
PACK-F2F2F2  UNIT-F2    DATE-07/10/75    ID-
DEVICE CAPACITY-400
AVAILABLE SPACE ON PACK
   LOCATION    TRACKS
      008         161

SEQ  FILE   RE-  FILE   FILE  REC KEY  KEY   DATA    FILE  FILE  RECORD  RECORDS  OCL SIZE NEXT AVAIL     NEXT AVAIL
NUM  NAME   TAIN DATE   TYPE  LEN LEN  LOC   START   LOC   TRACKS COUNT   AVAIL    PARAMETER RECORD        KEY

001 DIRF12   T 09/29/71  D 00124                    400   006      297  ******          6T     ******
002 DIR12A   T 07/10/75  D 00124                    394   006      297  ******          6T     ******
003 IND03R   P 07/10/75  S 00133                    374   020       28     895         20T 374/14/141
004 IND05B   P 07/10/75  S 00063                    339   035       52    3361         35T 339/12/205
005 SEQ09B   P 07/10/75  S 00512                    329   010       61      59         10T 334/02/001
006 IND03C   P 07/10/75  S 00133                    309   020       28     895         20T 309/14/141
007 IND05C   P 07/10/75  S 00063                    274   035       52    3361         35T 274/12/205
008 SEQ09C   P 07/10/75  S 00512                    264   010       61      59         10T 269/02/001
009 SEQ11C   P 07/10/75  S 00128                    258   006       99     189          6T 260/01/129
010 SEQ11C   P 10/30/71  S 00128                    189   007       99     237          7T 191/01/129
011 SEQ11C   P 10/01/71  S 00128                    196   012       98     478         12T 198/01/001
012 IND08C   T 07/10/75  S 00002                    255   003       54    9162          3T 255/00/109
013 DIR12C   T 07/10/75  D 00124                    249   006      297  ******          6T     ******
014 INDF07   P 07/10/75  S 00100                    247   002       71      51         71R 248/03/189
015 SEQF09   P 07/10/75  S 00512                    237   010        0     120         10T 237/00/001
016 INDF08   T 07/10/75  S 00002                    234   003       54    9162          3T 234/00/109
017 INDF10   T 07/10/75  S 04096                    208   026       38       1         38R 233/08/001
```

Figure 15. VTOC Printout Example

```
PACK-D1D1D1  UNIT-D1    DATE-07/09/75    ID-FIRSTPACK
NO. OF ALTERNATE TRACKS AVAILABLE-40
AVAILABLE SPACE ON PACK
   LOCATION    TRACKS
    002/00     0364
    021/19     0821
    063/02     0756
    101/01     0599
    131/10     0112
VTOC SIZE 1000 FILES

SEQ  FILE   RE-  FILE   FILE  REC KEY  KEY   DATA    FILE  FILE  RECORD  RECORDS  OCL SIZE NEXT AVAIL     NEXT AVAIL
NUM  NAME   TAIN DATE   TYPE  LEN LEN  LOC   START   LOC   TRACKS COUNT   AVAIL    PARAMETER RECORD        KEY

001 DIRF12   T 07/09/75  D 00124                    147/06 0002      198  *****           2T     *****
002 DIR12A   T 07/09/75  D 00124                    141/02 0006      594  *****           6T     *****
003 DIR12C   T 07/09/75  D 00124                    137/02 0006      594  *****           6T     *****
004 INDF03   P 07/09/75  I 00133 05 00104 001/02   001/00 0020       28    1635         20T 001/02/15/141 001/00/02/001
005 INDF05   P 07/09/75  I 00063 29 00063 020/17   020/04 0035       52    4239         35T 020/17/13/205 020/04/08/256
006 INDF07   P 07/09/75  I 00100 08 00099 063/01   063/00 0002       65      57         65R 063/01/26/101 063/00/04/025
             VOL SEQ NUM - 01       LOKEY-                                 HIKEY-00000005
                                                                                  0000006F

007 INDF07A  P 07/09/75  I 00100 08 00099 144/07   144/06 0005       71     420          5T 144/07/28/189 144/06/04/097
008 INDF08   T 07/09/75  I 00002 02 00002 101/00   100/18 0003       54    3894          3T 101/00/01/109 100/18/02/073
009 INDF10   T 07/09/75  I 04096 07 04096 144/00   143/19 0004        8       1          8R 144/02/33/001 143/19/01/089
             VOL SEQ NUM - 01       LOKEY-                                 HIKEY-0000008

010 INDF10A  T 07/09/75  I 04096 07 04096 141/09   141/08 0035        0     102         35T 141/09/01/001 141/08/01/001
011 IND03A   P 07/09/75  I 00133 05 00104 146/08   146/06 0020       28    1635         20T 146/08/15/141 146/06/02/001
012 IND03C   P 07/09/75  I 00133 05 00104 140/04   140/02 0020       28    1635         20T 140/04/15/141 140/02/02/001
013 IND05A   P 07/09/75  I 00063 29 00063 145/04   144/11 0035       52    4239         35T 145/04/13/205 144/11/08/256
014 IND05C   P 07/09/75  I 00063 29 00063 139/00   138/07 0035       52    4239         35T 139/00/13/205 138/07/08/256
015 IND08A   T 07/09/75  I 00002 02 00002 144/05   144/03 0003       54    3894          3T 144/05/01/109 144/03/02/073
016 IND08C   T 07/09/75  I 00002 02 00002 137/10   137/08 0003       54    3894          3T 137/10/01/109 137/08/02/073
017 OUT01    T 07/09/75  S 00096                    166/00 0020        1    2559         20T 166/00/01/097
018 OUT02    T 07/09/75  S 00255                    156/00 0020        1     962         20T 156/00/01/256
019 OUT03    T 07/09/75  S 00511                    165/00 0020        1     479         20T 165/00/02/256
020 OUT04    T 07/09/75  S 00096                    155/00 0020        1    2559         20T 155/00/01/097
021 OUT05    T 07/09/75  S 00767                    164/00 0020        1     319         20T 164/00/03/256
022 OUT06    T 07/09/75  S 00020                    154/00 0020        1   12287         20T 154/00/01/021
023 OUT07    T 07/09/75  S 00065                    163/00 0020        1    3779         20T 163/00/01/066
024 OUT08    T 07/09/75  S 00256                    153/00 0020        1     959         20T 153/00/02/001
025 OUT09    T 07/09/75  S 00257                    162/00 0020        1     955         20T 162/00/02/002
026 OUT10    T 07/09/75  S 04096                    152/00 0020        1      59         20T 152/00/17/001
027 OUT11    T 07/09/75  S 00001                    161/00 0020        1  245759         20T 161/00/01/002
028 OUT12    T 07/09/75  S 00100                    151/00 0020        1    2456         20T 151/00/01/101
029 OUT13    T 07/09/75  S 00010                    160/00 0020        1   24575         20T 160/00/01/011
030 OUT14    T 07/09/75  S 00127                    150/00 0020        1    1934         20T 150/00/01/128
031 OUT15    T 07/09/75  S 00128                    159/00 0020        1    1919         20T 159/00/01/129
032 OUT16    T 07/09/75  S 00128                    149/00 0020        1    1919         20T 149/00/01/129
033 OUT17    T 07/09/75  S 01024                    158/00 0020        1     239         20T 158/00/05/001
034 OUT18    T 07/09/75  S 00032                    148/00 0020        1    7679         20T 148/00/01/033
035 OUT19    T 07/09/75  S 00064                    157/00 0020        1    3839         20T 157/00/01/065
036 SEQF09   P 07/09/75  S 00512                    131/00 0010        0     240         10T 131/00/01/001
037 SEQF11   P 07/09/75  S 00128                    147/08 0012       97    1055         12T 147/09/01/129
038 SEQ09A   P 07/09/75  S 00512                    143/09 0010       61     179         10T 143/11/27/001
039 SEQ09C   P 07/09/75  S 00512                    137/17 0010       61     179         10T 137/19/27/001
040 SEQ11A   P 07/09/75  S 00128                    143/03 0006        0     576          6T 143/03/01/001
041 SEQ11C   P 07/09/75  S 00128                    137/11 0006       99     477          6T 137/12/02/129
```

Figure 16. 3340 Main Data Area VTOC Printout

MEANING OF VTOC INFORMATION

Heading	Meaning
PACK-name	Name of the disk.
Unit-code	Location of the disk containing the VTOC information
DATE-xx/xx/xx	Program level date.
ID-characters	Additional disk identification (if any).
NUMBER OF ALTERNATE TRACKS AVAILABLE-number	Number of alternate tracks available for assignment. Main data area only.
TRACKS WITH ALTERNATE ASSIGNED	Address of primary tracks that have been assigned an alternate. Main data area only.
DEFECTIVE ALTERNATE TRACKS	Address of the alternate tracks that are defective. Main data area only.
DEVICE CAPACITY-number	Disk capacity (number of tracks). Simulation area only.
LIBRARY EXTENT	Boundary of libraries on the disk. (If the simulation area contains no libraries, these headings are not printed.)
START	Track on which library begins. ⎫ If the simulation area contains both source and object library, START refers to beginning of source library and END refers to end of object library.
END	Track on which library ends. ⎭
EXTENDED END	Object library only (simulation area only). Track on which extension to library ends. When object library is full, temporary entries can be placed in space following end of library, provided that space is available.
AVAILABLE SPACE ON PACK	Available disk areas.
LOCATION	First track in available area (simulation area). First cyclinder/track in available area (main data area).
TRACKS	Number of tracks available.
VTOC SIZE {50 FILES / 1000 FILES}	Maximum number of entries in VTOC.
SEQ NUM	Line number.
FILE NAME	Name that identifies file in VTOC.
RETAIN	File designation: P = Permanent T = Temporary S = Scratch (simulation area only)
FILE DATE	Date given the file when file was placed on disk.

Heading	Meaning
FILE TYPE	File type: I = indexed S = sequential D = direct * = file used by spooling
REC LEN	Number of characters in each record in file.
KEY LEN	Number of characters in each record key (indexed file only).
KEY LOC	Position in record occupied by last character of record key (indexed files only).
DATA START	Disk area reserved for indexed files only. DATA START is the first main data area cylinder/track of the area. This refers to the data portion of the file.
FILE LOC	First track used by the file. For simulation area files, refers to a track number. For main data area files, refers to a cylinder/track number.
FILE TRACKS	Number of tracks allocated to the file.
RECORD COUNT	Total number of records currently in the file.
RECORDS AVAIL	Number of records that can be added to the file. For indexed files, more records may be added than the number indicated in this field.
OCL SIZE PARAMETER	Parameter used on OCL statement when file was created. T = tracks R = records

Heading	Meaning
NEXT AVAIL RECORD	Beginning location of next available record in file. For simulation area, location is track, sector, and position within sector. For main data area, location is cylinder, track, fixed record, and position within record. Example: 099/18/006 = track 99, sector 18, positions 6.① 050/02/12/006 = cylinder 50, track 2, fixed record 12, position 6.①
NEXT AVAIL KEY	Indexed files only. Beginning location of next available record key in index portion of file. For simulation area, location is track, sector, and position within sector. For main data area, location is cylinder, track, fixed record, and position within record. Main data area only. Example: 090/10/006 = track 90, sector 10, positions 6.② 052/03/10/006 = cylinder 52, track 3, fixed record 10, position 6.②
VOL SEQ NUM	VOL SEQ NUM applies to multivolume files only. It indicates the order of the disk as it relates to the other disks containing the remaining portion of the file. Main data area only.
LOKEY	The high key from the previous volume. This field will be blank for the first volume of a multivolume file. Main data area only.
HIKEY	The highest key that can be put on the multivolume indexed file. Main data area only.

① If the first byte of the next available record occurs in the next track after the end track of DATA START END or if there is no room for additional index area, then this field will contain ****.

② If the first byte of the next available key occurs in the next track after the end track of INDEX START END, or there is no room for additional index area, then this field will contain ****.

FILE INFORMATION ONLY

The parameter LABEL-filename or LABEL-'filenames' means to print certain file information from the VTOC. For one file, use LABEL-filename; for two files, use LABEL-'filename,filename'; and so on. (Use the names that identify the files in the VTOC.) You can list 20 filenames for a program run. The statement length, however, is restricted to 96 characters.

The program prints the file information for each of the files you list. This is the information described for the headings PACK name and FILE LABEL under *Meaning of VTOC Information.*

If the program needs more than one page to list the file information, it prints headings for the file information at the top of each new page.

OCL Considerations

The following OCL statements are used to load the file and volume label display program.

```
// LOAD $LABEL,code
// RUN
```

The code you supply depends on the location of the simulation area containing the utility program. The codes are R1, F1, R2, and F2.

EXAMPLE

Printing VTOC Information for Two Files

Figures 17 and 18 are examples of the OCL statements and utility control statements needed to print VTOC information for two files.

```
1   4   8   12  16  20  24  28  32  36
/&
// LOAD $LABEL,F1
// RUN
```

Explanation:

The file and volume label display program is loaded from simulation area F1 on drive 1.

Figure 17. OCL Load Sequence for File and Volume Label Display

```
1   4   8   12  16  20  24  28  32  36  40  44  48  52  56  60  64
// DISPLAY UNIT-D1,LABEL-'BILLNG,INVO1'
// END
```

Explanation:

The files for which information is printed are named BILLNG and INVO1 (LABEL-'BILLNG, INVO1' in DISPLAY statement). They are located on main data area D1 on drive 1 (UNIT-D1).

Figure 18. Utility Control Statements for Printing VTOC Information for Two Files

File Delete Program—$DELET

The file delete program has four uses:

- Removing all files from a disk.
- Removing only the files you name.
- Scratching file references in the volume table of contents (VTOC). Deleting files frees the space they occupy for use by new files.
- Formatting a simulation or main data area.

The program may be used on temporary, scratch and permanent files. To delete permanent files, you must use the file delete program. You can scratch temporary files by using the file delete program or by changing the file designation from temporary to scratch (using the OCL keyword RETAIN) when you use the file.

The control statements you supply for the file delete program depend on the function to be performed.

When the REMOVE statement is used, files are erased from the VTOC. The REMOVE statement can also be used to erase files from the disk. When the SCRATCH statement is used for a file in the main data area, it performs the same function as REMOVE. The SCRATCH statement does not erase files from the simulation areas. It changes their designation to scratch (S) in the VTOC. By doing this, the program makes the areas that contain the files available for other files or for system programs.

The FORMAT statement is used to free all allocated space that does not contain files, libraries, or system areas. This statement is used when you suspect that a system failure or an inadvertent re-IPL might have left space allocated, but not actually being used, on the data module.

CONTROL STATEMENT SUMMARY

Use	Control Statements[1]
Scratch all files in the VTOC.	// SCRATCH PACK-name, UNIT-code, LABEL-VTOC // END
Scratch only one file in the VTOC.	// SCRATCH PACK-name, UNIT-code, LABEL-filename, DATE-date[2] // END
Scratch multiple files in the VTOC	// SCRATCH PACK-name, UNIT-code, LABEL- {filename / 'filenames'} // END
Remove all files from disk	// REMOVE PACK-name, UNIT-code, LABEL-VTOC, DATA- {NO / YES} // END
Remove only the files named from disk	// REMOVE PACK-name, UNIT-code, LABEL- {filename / 'filenames'} DATE-date, DATA[2]- {NO / YES} // END
Free allocated but unused space	// FORMAT PACK-name, UNIT-code // END

---

[1] For each use, the program requires the statements in the order they are listed: SCRATCH, END, or REMOVE, END, or FORMAT, END.

[2] Use this form of the SCRATCH or REMOVE statement when two or more files have the same name and you want to delete one of them.

[3] Use this control statement when you suspect that a system failure or an inadvertent re-IPL may have left space allocated, but not actually being used, on the disk.

PARAMETER SUMMARY

PACK-name	Name of the disk.	
UNIT-code	Location of the disk. Possible codes are R1, F1, R2, F2, D1, D2.	
LABEL-VTOC	Scratch or remove all files from the VTOC.	
LABEL-filename	Scratch or remove only the file named in the VTOC.	Use names that identify files in VTOC.①
LABEL-'filename,filename,...	Scratch or remove only the files named in the VTOC.	
DATE-date	Date of the file being deleted. Date must be a 6-digit number. Example: DATE-032076 means March 20, 1976.	
DATA - {NO / YES}	Delete files from disk as well as VTOC.	

① These are the names you gave the files when you placed them on disk.

PARAMETER DESCRIPTIONS

PACK Parameter

The PACK parameter (PACK-name) tells the program the name of the disk that contains the files being deleted. The name you supply in this parameter is the one written on the main data area by the disk initialization program.

For a simulation area it is the name assigned by the simulation area program $SCOPY.

The file delete program compares the name in the PACK parameter with the name on the disk to ensure they match. In this way, the program ensures that it is using the right disk.

UNIT Parameter

The UNIT parameter (UNIT-code) tells the program the location of the disk containing the files being deleted. Codes for the possible locations are R1, F1, R2, F2, D1, and D2.

LABEL Parameter

The LABEL parameter identifies the files you want to delete from the disk. Its form depends on the files you are deleting:

Form	Files Deleted
LABEL-VTOC	All of them.
LABEL-filename	Only the file that is named. The name can apply to more than one file. If it does, all of those files are deleted unless you use a DATE parameter to identify a particular one.
LABEL-'filename,filename,...'	Only the files that are named. A name can apply to more than one file. If it does, all of those files are deleted. You can list as many filenames as the statement can hold; the statement length, however, is restricted to 96 characters. Additional REMOVE or SCRATCH statements may be used for additional filenames.

DATE Parameter

The DATE parameter can be used only with LABEL-filename. The DATE parameter (DATE-date) applies to two or more files that have the same name. It tells the program the date of the one you want to delete.

Every file on disk has a date, which is given to the file at the time it is created. When two or more files have the same name, the dates are used to distinguish one file from another.

If the pack has more than one file with the name you list in the LABEL parameter, they will all be deleted unless you use the DATE keyword and parameter to indicate a particular file. If the DATE keyword is used, only one filename can be given in the LABEL parameter for that control statement.

The date is a 6-digit number: two digits for day, two for month, and two for year. Day, month, and year can be in one of two formats as specified at system generation time: (1) month, day, year, and (2) day, month, year. For example, 021676 and 160276 both mean February 16, 1976.

In the DATE parameter, be sure to specify day, month, and year in the same order as they were specified when you placed the file on disk.

DATA Parameter

The DATA parameter lets you remove the files specified directly from the disk as well as from the VTOC.

If YES is coded in this parameter, the file specified is removed from the disk and any reference to it in the VTOC is removed. In addition, a message is printed on the system log device for each file removed from the disk in this format:

```
'DATA REMOVED FOR FILE XXXXXX
DATE 000000'
```

DATA-YES should be used only if file security is required. The time needed to remove the data is much greater than the time needed to remove the VTOC entry.

If NO is coded in this parameter, the file specified is not removed from the disk. However, any reference to it in the VTOC is removed. If this parameter is not used, DATA-NO is assumed.

OCL CONSIDERATIONS

The following OCL statements are needed to load the file delete program:

```
// LOAD $DELET,code
// RUN
```

The code you supply depends on the location of the simulation area containing the utility program. The codes are R1, F1, R2, and F2.

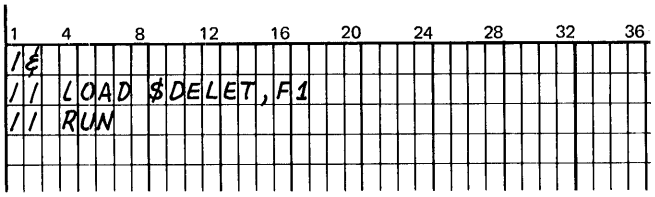
EXAMPLES

Deleting One of Several Files Having the Same Name

Figures 19, 20, and 21 are examples of the OCL statements and utility control statements needed to delete one of several files having the same name as described in the following situation.

Situation

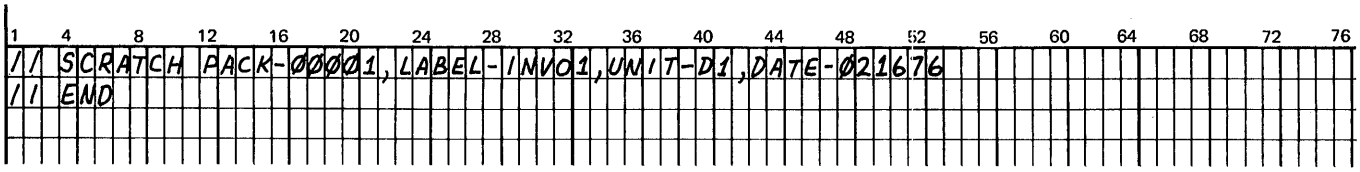
Assume that three files in the main data area have the same name: INVO1. The dates of these files are 2/16/76, 2/18/76, and 1/15/76. You want to delete the version dated 2/16/76.



Explanation:

File delete program is loaded from simulation area F1 on drive 1.

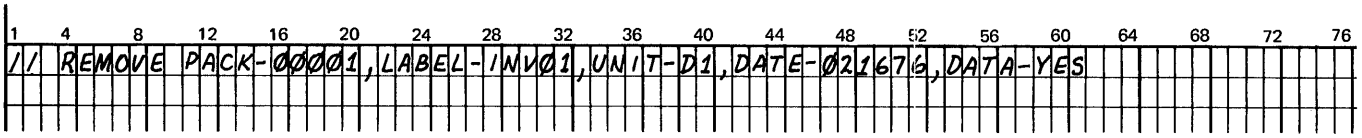
Figure 19. OCL Load Sequence for File Delete



Explanation:

- Main data area that contains the file being deleted is named 00001 (PACK-00001 in SCRATCH statement).
- Because two other files have the name INVO1, the date (021676) is needed to cornplete the identification of the file you want to delete (LABEL-INVO1 and DATE-021676).
- The main data area containing the file to be deleted is on drive 1 (UNIT-D1).

Figure 20. Utility Control Statements to Delete One Version of a File



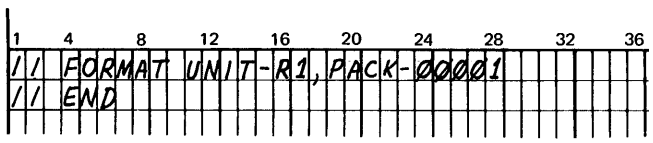
Explanation:

- A REMOVE statement is used instead of a SCRATCH statement.
- Main data area that contains the file being deleted is named 00001 (PACK-00001 in REMOVE statement).
- Because two other files have the name INV01, the date (021676) is needed to complete the identification of the file you want to delete (LABEL-INV01 and DATE-021676).
- The main data area containing the file to be deleted is on drive 1 (UNIT-D1).
- The YES specification in the DATA parameter deletes all data from the disk containing information on the specified file.

Figure 21. Utility Control Statement to Delete One Version of a File Using a REMOVE Statement

Freeing Allocated But Unused Space on a Disk

Figure 22 shows the FORMAT control statement. The following will free any areas on the simulation area (R1) that have been allocated but are not being used. This condition may exist following the abnormal termination (such as a power failure or re-IPL) of a program that was creating a file.



Explanation:

Free any allocated but unused space on the simulation area (R1) named 00001 (UNIT-R1).

Figure 22. Control Statements to Free Allocated But Unused Space on a Simulation Area

Dump/Restore Program—$DCOPY

The dump/restore program ($DCOPY) is a utility program used with the IBM System/3 Model 12 system control program. The $DCOPY program allows the user to copy or dump the entire contents of a disk onto tape. The tape then serves as a backup copy in case something happens to the information on the disk.

The program can restore the disk to its original contents at any time by transferring information back from the tape. Important disks, such as those containing libraries and permanent data files, are normally the ones copied. The tape contains a copy of the data on all tracks.

The program can also dump or restore the simulation areas using a 3741 diskette.

CONTROL STATEMENT SUMMARY

Uses	Control Statements①
Copy an entire disk to tape or restore an entire disk from tape.	// COPYPACK② {TO-code / FROM-code} [,PACK-name]
	// END③
	// COPYPACK {TO-code / FROM-code} [,PACK-name] [,SYSTEM- YES / NO] [,BACKUP- TAPE / 3741]
	// END

① Control statements are required in the order they are listed.

② There can be only one COPYPACK statement in a program.

③ END statement must appear only once in a program since it is a delimiter indicating end of job.

PARAMETER SUMMARY

COPYPACK Statement

Parameter	Meaning
FROM-code	Location of disk to be copied. Possible codes are F1, R1, F2, R2, D1, D2.
TO-code	Location of disk to receive the copy. Possible codes are F1, R1, F2, R2, D1, D2. See Figure 21 for relationship of FROM and TO locations.
PACK-name	Name of the main data area or simulation area being used.
SYSTEM-NO	The SYSTEM-NO parameter does not allow cylinder 0 IPL areas to be dumped or restored.
SYSTEM-YES	SYSTEM-YES specifies that the IPL areas on cylinder 0 are to be dumped or restored along with the specified simulation area.
BACKUP-TAPE	The BACKUP-TAPE parameter specifies that magnetic tape (3410-3411) is to be used for dump/restore.
BACKUP-3741	BACKUP-3741 specifies that the 3741 diskette is to be used to dump or restore the specified simulation area.

PARAMETER DESCRIPTIONS

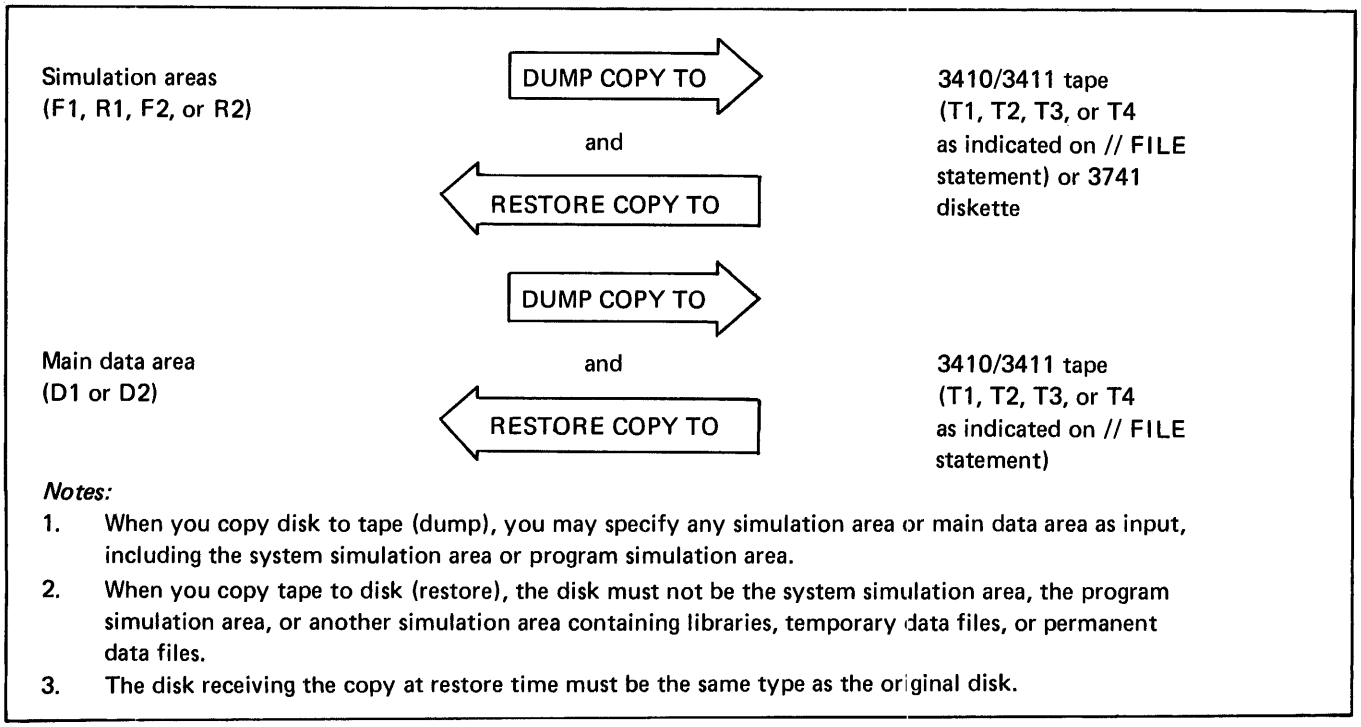
FROM and TO Parameters (COPYPACK)

The COPYPACK statement is used to copy information from disk to tape, tape to disk, disk to diskette, or diskette to disk.

The FROM parameter (FROM-code) indicates the location of the disk being copied. The TO parameter (TO-code) indicates the location of disk to receive the copy.

Codes for possible locations of FROM and TO parameters are R1, F1, R2, F2, D1, and D2.

See Figure 23 for the relationship of FROM and TO locations.



Notes:

1. When you copy disk to tape (dump), you may specify any simulation area or main data area as input, including the system simulation area or program simulation area.
2. When you copy tape to disk (restore), the disk must not be the system simulation area, the program simulation area, or another simulation area containing libraries, temporary data files, or permanent data files.
3. The disk receiving the copy at restore time must be the same type as the original disk.

Figure 23. Relationship of Disk to Tape Drives When Using $DCOPY

PACK Parameter (COPYPACK)

The pack name specified is checked against the actual name of the main data area or simulation area. A halt occurs if they are not the same. If the parameter is not used, no checking occurs.

SYSTEM Parameter (COPYPACK)

The SYSTEM parameter is an optional parameter used to specify whether cyclinder 0 IPL information is to be dumped or restored with the specified simulation area. SYSTEM-YES allows cylinder 0 to be dumped or restored to either tape or diskette. SYSTEM-NO does not allow cylinder 0 to be dumped or restored. The default is SYSTEM-NO.

BACKUP Parameter (COPYPACK)

The BACKUP parameter specifies which device (tape or diskette) is to be used for backup. Tape may be used to back up the main data area, simulation areas, and cylinder 0. The 3741 diskette can be used only to back up the simulation areas and cylinder 0. Also, the 3741 data set must be set for 128 byte records.

OCL CONSIDERATIONS

The $DCOPY utility requires the following OCL statements:

```
// LOAD $DCOPY, code

// FILE parameters

// RUN
```

The code identifying the location of the $DCOPY program can be R1, F1, R2, or F2.

FILE Statement Considerations

- The name of the file must always be BACKUP.
- When a 7-track tape is used for the dump/restore program, CONVERT-ON must be specified.
- The record format is always fixed length.
- The END position of the tape after processing always defaults to UNLOAD.
- The density parameter when restoring must be the same number as specified for the dump.
- The record length, if specified, is ignored since $DCOPY makes the record length equal to the block length.

For a detailed description of the FILE statement parameters, see *File Statement (Tape)* in Part 1 of this manual.

Notes:

1. The FILE statement is not required when copying from disk to diskette.
2. For multivolume tapes, see *Multivolume Tape Files* under *FILE Statement (Tape)* in Part 1 of this manual.

Statement Entries

Statement Entry	Considerations
// LOAD	None
$DCOPY	Name of dump/restore program.
code	Location of simulation area containing dump/restore program. Can be R1, F1, R2, or F2.
// FILE	None
NAME-filename	Filename entry must be BACKUP.
BLKL-block length	Block length and record length must be equal and one of the following values: *Note:* The tape record created is 2 bytes longer than specified since a 2-byte logical record number is appended to the tape record. Defaults are underlined.

Disk	Length in Bytes	Number of Tracks
Simulation area	<u>3072</u>	1/2 track
	6144	1 track
	12288	2 tracks
Main data area	<u>3072</u>	1/4 track
	6144	1/2 track
	12288	1 track
	24576	2 tracks

// RUN	None

For a detailed description of the OCL statements, see Part 1 of this manual.

Note: The rest of the FILE statement parameter is described by the TAPE FILE OCL statement.

Messages for DUMP/RESTORE

Note: The following messages are printed if the 1403 or 5203 is the logging device and is not allocated to the other program level.

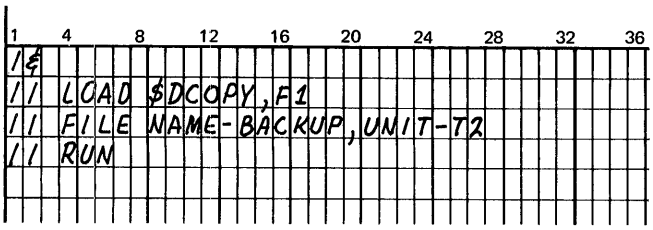
Message	Meaning
COPYPACK IS COMPLETE	This message is printed when the specified pack has been dumped to tape or when the tape has been restored to disk.
N TRACKS NOT RESTORED AT {CC/SS / CCC/HH/RR}	This message is printed when tracks have not been restored on the simulation area or main data area. N = the number of tracks not restored. CC/SS is the disk address for a simulation area. CCC/HH/RR is the disk address for a main data area.
NN TAPE ERRORS OCCURRED PACK IS NOT COMPLETELY RESTORED.	This message is printed when tape errors have occurred or the restored pack has missing data. NN = the number of tape errors. See previous messages for location of tracks not restored.

EXAMPLES

The parameters of the FILE statement vary depending upon whether the copy is to or from the tape.

FILE Statement: From Disk to Tape

Only required parameters are included in this example. See *OCL Considerations* for a listing of possible parameters.

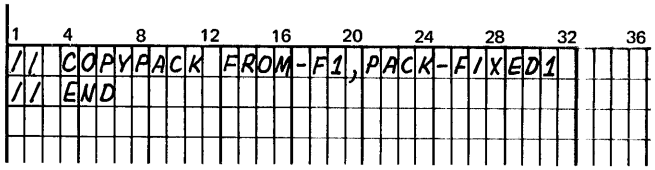


Explanation:

- The dump/restore program is loaded from simulation area F1 on drive 1.
- The file name is always BACKUP.
- The copy goes to tape unit 2.
- Tape unit 2 is a 9-track drive.

Control Statements

The following control statements show the use of all possible parameters:

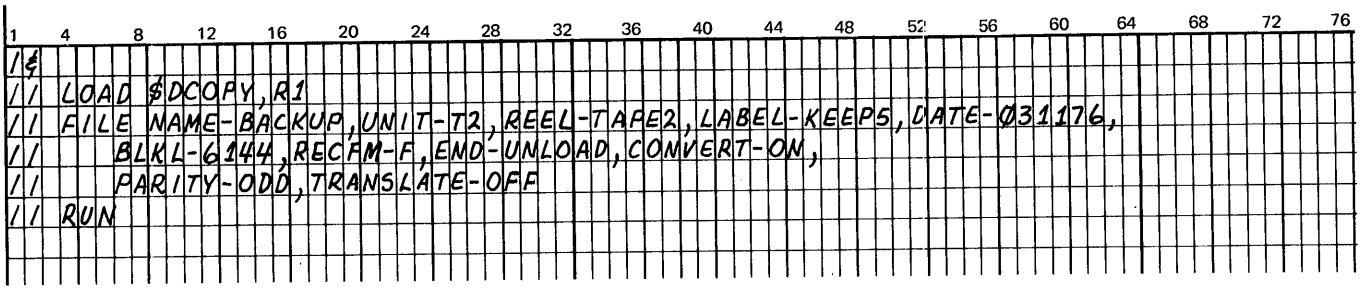


Explanation:

- The COPYPACK statement tells the program to copy an entire disk to tape.
- The copy is from the simulation area F1 on drive 1 (FROM-F1).
- FIXED1 is the name of the simulation area being used (PACK-FIXED1). The program verifies that the specified data module is mounted.

FILE Statement: From Tape to Disk

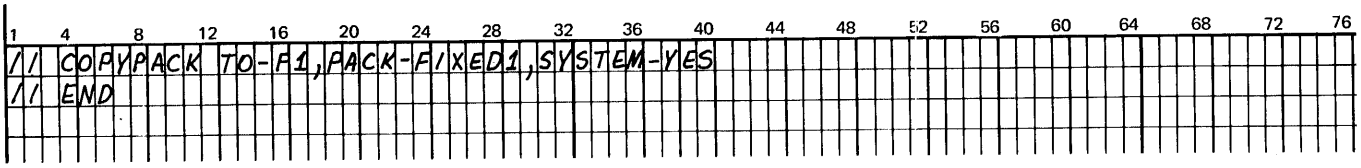
All possible parameters are included in this example.



Explanation:

- The dump/restore program is loaded from the simulation area R1 on drive 1.
- The file name is always BACKUP.
- Tape unit 2 contains the disk copy.
- Tape unit 2 is a 7-track drive.
- TAPE2 is the label of the tape volume.
- KEEP5 is used in the header label.
- The date is March 11, 1976.
- Block length is 6144.
- CONVERT-ON indicates data conversion.
- END, PARITY, and TRANSLATE parameters given are the same as the default values.

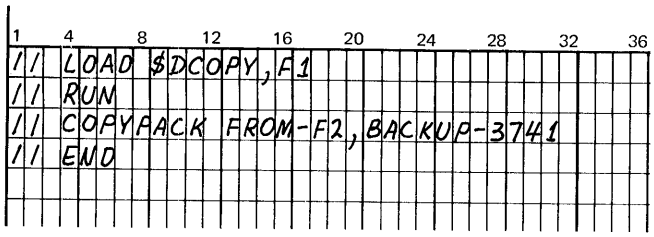
The following control statements show the use of all possible parameters:



Explanation:

- The COPYPACK statement tells the program to copy an entire tape to simulation area F1 (TO-F1).
- The statement restores cylinder 0 IPL of the data module along with simulation area F1 on drive 1.
- FIXED1 is the name of the simulation area being used (PACK-FIXED1). The program verifies that the proper pack is mounted.

Control Statement: From Disk to Diskette



Explanation:

- The dump/restore program is loaded from the simulation area F1 on drive 1.
- The COPYPACK statement tells the program to copy the simulation area F2 (FROM-F2) to the 3741 (BACKUP-3741).
- It will take approximately 11 diskettes to contain the copy from simulation area F2.
- The record length on the 3741 diskette must be 128.

Programming Considerations

When dumping from one of the simulation areas to diskette, it is recommended that you put the 3741 online in Mode 3. (Modes 1, 2, and 5 will result in extent error conditions at the end of each diskette.) See note.

When restoring from diskette to one of the simulation areas, it is recommended that you put the 3741 online in Mode 3 or Mode 5. If the 3741 is put online in Mode 1, $DCOPY will go to end of job at the end of the first diskette. If the 3741 is put online in Mode 2, the operator will have to put the 3741 online after each diskette is read. See note.

The COPYPACK IS COMPLETE message will be logged at successful completion of $DCOPY. If this message is not logged after restoring to disk, the simulation area copied to will not be usable.

Note: Refer to *IBM System/3 3741 Reference Manual,* GC21-5113, for further explanation of the 3741 modes of operation.

Copy/Dump Program—$COPY

The copy/dump program has three general uses. The control statements you must supply depend on the program use.

Program Use	Situation
Copy the entire contents of a simulation area to another simulation area, or copy the entire contents of a main data area to another main data area.	Provide a reserve disk in case something happens to the original disk. Important disks, such as those containing your libraries and permanent data files, are normally the ones you would copy.
Copy all or part of a data file from disk, diskette, tape, or cards, to disk, diskette, tape, or cards. (See note.) *Note:* A diskette file cannot be copied to another diskette.	Any of the following: • Provide a reserve (backup) file in case something happens to the original file. • Move a file to a larger disk area. • Reorganize the data portion of an indexed file. (Data in the copy of the file is reorganized; the original file is unchanged.) • Delete records from a file. (Records are omitted from the copy of the file; the original file remains unchanged.) • Create disk, diskette, card, or tape files. • Create indexed disk files from sequential files. • Copy card decks to disk, diskette, or tape.
Print all or part of a data file.	Provide a printed copy of the records in a file, perhaps for use in checking the records for errors.
Recover data by means of physical address.	Provides a way to recover data lost due to abnormal termination of a job.

The OCL sequence used to load the program describes the disk or tape file being copied or printed. If you are copying the file to disk or tape, the file being created must also be described in the OCL sequence.

No OCL FILE statements are required for card, printer, or diskette files. (When you are copying card, printer, or diskette files, you describe the input and output in the // COPYFILE control statement.)

Note: When you are copying large indexed files, you may realize a time savings by specifying reorganization if the data records are not in the same sequence as the keys in the index portion of the file.

CONTROL STATEMENT SUMMARY

Uses①	Control Statements②
Copy an entire disk	// COPYPACK FROM-code,TO-code,PACKIN-name,PACKO-name // END

Copy a data file	// COPYFILE {OUTPUT-|OUTPTX-} (FILE | DISK | MFCU | MFCU1 | MFCU2 | 1442 | 3741) ,INPUT- (MFCU | MFCU1 | MFCU2 | 3741 | 1442) ,LENGTH-number,④
	{DELETE-|OMIT-} 'position,character',REORG- {NO | YES} ,WORK- {YES | NO}⑤
	// END

Copy and print a data file	// COPYFILE {OUTPUT-|OUTPTX-} (BOTH | 'PRINT,MFCU' | 'PRINT,MFCU1' | 'PRINT,MFCU2' | 'PRINT,1442' | 'PRINT,3741') ,INPUT- (MFCU | MFCU1 | MFCU2 | 3741 | 1442) ,LENGTH-number,④
	{DELETE-|OMIT-} 'position,character',REORG-YES,WORK- {YES | NO}⑤
	// END

Copy a data file, but print only a part of the file	// COPYFILE {OUTPUT-|OUTPTX-} (BOTH | 'PRINT,MFCU' | 'PRINT,MFCU1' | 'PRINT,MFCU2' | 'PRINT,1442' | 'PRINT,3741') ,INPUT- (MFCU | MFCU1 | MFCU2 | 3741 | 1442) ,LENGTH-number,④
	{DELETE-|OMIT-} 'position,character',REORG-YES③,WORK- {YES | NO}⑤
	// SELECT KEY,FROM-'key' [,TO-'key']
	// SELECT RECORD,FROM-number [,TO-number]
	//SELECT PKY,FROM-'key'[,TO-'key']
	// END
	Only one SELECT statement for each COPYFILE statement

Print an entire data file	// COPYFILE {OUTPUT-|OUTPTX-} PRINT,INPUT- (MFCU | MFCU1 | MFCU2 | 3741 | 1442) ,LENGTH-number④ // END

Note: MFCU and MFCU1 refer to the MFCU hopper 1 (primary).

Uses	Control Statements

Print only a part of a data file	// COPYFILE {OUTPUT- / OUTPTX-} PRINT,INPUT- {MFCU / MFCU1 / MFCU2 / 3741 / 1442} ,LENGTH-number④
	// SELECT KEY,FROM-'key' [,TO-'key']
	// SELECT RECORD,FROM-number [,TO-number]
	// SELECT PKY,FROM-'key' [,TO-'key']
	// END
	Only one SELECT statement for each COPYFILE statement

Print and copy a part of a data file	// COPYFILE {OUTPUT- / OUTPTX-} {BOTH / 'PRINT,MFCU' / 'PRINT,MFCU1' / 'PRINT,MFCU2' / 'PRINT,1442' / 'PRINT,3741'} ,INPUT- {MFCU / MFCU1 / MFCU2 / 3741 / 1442} ,LENGTH-number④
	WORK- {YES / NO} ⑤
	// SELECT KEY,FROM-'key' [,TO-'key'],FILE-YES
	// SELECT RECORD,FROM-number [,TO-number],FILE-YES
	// SELECT PKY,FROM-'key' [,TO-'key'],FILE-YES
	// END
	Only one SELECT statement for each COPYFILE statement

Copy part of a data file	// COPYFILE {OUTPUT- / OUTPTX-} {FILE / DISK / MFCU / MFCU1 / MFCU2 / 1442 / 3741} ,INPUT- {MFCU / MFCU1 / MFCU2 / 3741 / 1442} ,LENGTH-number④,WORK- {YES / NO} ⑤
	// SELECT KEY,FROM-'key' [,TO-'key'],FILE-YES
	// SELECT RECORD,FROM-number [,TO-number],FILE-YES
	// SELECT PKY,FROM-'key' [,TO-'key'],FILE-YES
	// END
	Only one SELECT statement for each COPYFILE statement.

Notes:

1. MFCU and MFCU1 refer to the MFCU hopper 1 (primary).
2. MFCU defaults to MFCU1.

Uses	Control Statements
Build an indexed file from a sequential file	// COPYFILE {OUTPUT-|OUTPTX-} {FILE|DISK|BOTH} ,INPUT- {MFCU|MFCU1|MFCU2|3741|1442} ,LENGTH-number④
	// KEY LENGTH-number,LOCATION-number
	// END

[1] The program uses include the possible combination of copying and printing files.

Recover data by physical address (simulation area)	// COPYFILE {OUTPUT-|OUTPTX-} {FILE|DISK|BOTH}
	// ACCESS FROM-code,CYLINDER-number,SECTOR-number, DISP-number,RECL-number
	// SELECT RECORD,FROM-number,TO-number,FILE-YES
	// END
Recover data by physical address (main data area)	// COPYFILE {OUTPUT-|OUTPTX-} {FILE|DISK|BOTH}
	// ACCESS FROM-code,CYLINDER-number,TRACK-number, SECTOR-number,DISP-number,RECL-number
	// SELECT RECORD,FROM-number,TO-number,FILE-YES
	// END

---

① The program uses include the possible combination of copying and printing files.

② For each use, the program requires the control statements in the order they are listed: COPYPACK,END; COPYFILE,END; COPYFILE,SELECT,END; COPYFILE,KEY,END; and COPYFILE,SELECT,KEY,END.

③ Applies only to indexed files. When OUTPUT-BOTH is specified, REORG-YES is required.

④ Optional — the record length defaults to 96 when the 3741 is used for input or output if LENGTH is not specified.

⑤ Optional — must have simulation turned off on D2, then copies from one data module on D2 to a different data module on D2.

Note: MFCU defaults to MFCU1.

PARAMETER SUMMARY

COPYPACK Statement

FROM-code	Location of disk to be copied. Possible codes are R1, F1, R2, F2, D1, and D2.
TO-code	Location of disk to contain the copy. Possible codes are R1, F1, R2, F2, D1, and D2.
PACKIN-name	Volume identification (name) of FROM disk.
PACKO-name	Volume identification (name) of TO disk.

COPYFILE Statement

OUTPUT-FILE	Copy the file to the device (tape or disk) defined in the COPYO FILE statement.① (Interchangeable with OUTPUT-DISK.)
OUTPUT-DISK	Same as OUTPUT-FILE.
OUTPUT-{MFCU, MFCU1, MFCU2, 1442, 3741}	Copy the file to the device specified. When this parameter is used, a COPYO FILE statement must not be used.

OUTPUT-PRINT

Print the entire file or only part of the file.①

OUTPUT-BOTH③

Copy the file from one device to another or from one area to another on the same disk.① Also print the entire file or only part of it.

OUTPUT-{'PRINT,MFCU' / 'PRINT,MFCU1' / 'PRINT,MFCU2' / 'PRINT,1442' / 'PRINT,3741'}

Copy the file to the device specified. Also print the entire file or only part of it. When this parameter is used, a COPYO FILE statement must not be used.

OUTPTX-{PRINT / BOTH / 'PRINT,MFCU' / 'PRINT,MFCU1' / 'PRINT,MFCU2' / 'PRINT,1442' / 'PRINT,3741'}

Printed output will be displayed in hexadecimal values. If one of the card devices or 3741 is used, then a COPYO FILE statement must not be used.

INPUT-{MFCU / MFCU1 / MFCU2 / 3741 / 1442}

Copy the file from the device specified. If this keyword is used, then a COPYIN file statement must not be used.

LENGTH-number

Identifies the record length of a file on a diskette. Number must be an integer from 1 to 128. If this keyword is not specified, the record length defaults to 96. If used with a device other than a 3741, this keyword is ignored.

DELETE-'position,character'
-or-
OMIT-'position,character'

These parameters are optional. It means that all records with the specified character in the specified record position are deleted. DELETE causes deleted records to be printed. DELETE cannot be used with direct files. OMIT causes deleted records not be printed. Position can be any position in the record (the first position is 1, second 2, and so on). The maximum position is 65535.

REORG-NO②

Indexed files only. Copy records in the same way as they are organized in the original file (the file from which the records are copied).

REORG-YES②③

Indexed files only. Reorganize the records so that the records in the data portion of the file are in the same order as their keys are listed in the index.

WORK-YES

This parameter is required when a file is copied from one data module on drive 2 to another data module to be placed on drive 2.

SELECT Statement

{KEY / PKY} ,FROM-'key'	Indexed files only. Print and copy only the part of the file from the record key that is specified in the FROM parameter to the end of the file.
{KEY / PKY} ,FROM-'key',TO-'key'	Indexed files only. Print and copy only the part of the file between the two record keys that are specified in the FROM and TO parameters (including the records indicated by the parameters). To print and copy only one record, make the FROM and TO record keys the same.
RECORD,FROM-number	Print and copy only the part of the file from the relative record number specified in the FROM parameter to the end of the file.
RECORD,FROM-number, TO-number	Print and copy only the part of the file between the relative record numbers indicated by the parameters (including the records indicated by the parameter). To print and copy only one record, the FROM and TO relative record numbers should be the same. Record number may be from 1 to 16777215.
FILE-YES	Only selected records are copied to the files named in the COPYO FILE statement, or the device specified in the OUTPUT keyword parameter of the // COPYFILE control statement, when selected records are to be copied to the 3741 or a card device. The file is sequential if no // KEY statement is provided. When // KEY statement is used, the output is an indexed file if the device on the COPYO FILE statement is a disk.
FILE-NO	Only selected records are printed. If copying, all records are copied. OUTPUT-PRINT or OUTPUT-BOTH must be specified if FILE-NO is specified. If OUTPUT-BOTH is specified, selected records are printed and the entire file is copied to the file named in the COPYO FILE statement, or the device specified in the OUTPUT keyword parameter of the // COPYFILE control statement, when selected records are to be copied to the 3741 or a card device. If OUTPUT-PRINT is specified, selected records are printed only.

KEY Statement

LENGTH-number	Identifies the length of the key field. Key length may be 1-29.
LOCATION-number	The starting location in the input record that the key field is to be extracted from. Location may be from 1 to 65525.

ACCESS Statement

FROM-code	Location of data to be copied. Possible codes are R1, F1, R2, F2, D1, and D2.
CYLINDER-number	Cylinder location of start of data; for a main data area it may be a number from 0-166. For a simulation area copy, it may be a number from 0-202.
TRACK-number	Track location of start of data. It is a number from 0-19.
SECTOR-number	Sector number of start of copy. For a simulation area it may be a number from 0-47, for main data areas it may be a number from 1-48.
DISP-number	Displacement into sector of first good data to be recovered.
RECL-number	Record length of data to be recovered. Number may be between 1-65536.

---

(1) In the OCL load sequence, you indicate which file is to be copied or printed. For files being copied, you must also indicate whether the file is being copied from one device to another or from one location to another on the same disk, using the COPYIN and COPYO FILE statements. COPYIN and COPYO FILE statements are invalid for the 3741 printer and card devices. The INPUT and OUTPUT keywords in the // COPYFILE statements are used for the 3741 printer and card devices.

(2) REORG-NO is assumed if you omit the REORG parameter. When OUTPUT-BOTH is used for indexed files, REORG-YES is required.

(3) If halt UC3CCS occurs, indicating that there is not enough main storage available to execute the job, consider the following:

1. If you have OUTPUT-BOTH, change to OUTPUT-DISK or OUTPUT-FILE.
2. If you have REORG-YES, change to REORG-NO.
3. If running on a DPF system, use a larger program level if possible.

PARAMETER DESCRIPTIONS

FROM and TO Parameters (COPYPACK)

The FROM and TO parameters are used when the entire contents of one disk are copied onto another. They tell the program the locations of the two disks.

The FROM parameter (FROM-code) indicates the location of the disk you are copying. The TO parameter (TO-code) indicates the location of the disk that is to contain the copy. The FROM and TO codes must be for the same type of disk drive. You cannot copy a simulation area from or to a main data area.

Codes for the possible locations are R1, F1, R2, F2, D1, and D2.

Copying Entire Disk

When copying a disk, the copy/dump program transfers the contents of the disk to another disk. The contents of the two disks will be the same except for the disk names and alternate track information, which may be different.

The disk you are copying can contain libraries or data files or both. The disk that is to contain the copy must not contain libraries, temporary files, or permanent data files.

Until the entire contents of the disk are copied onto the new disk, portions of the new disk are changed to prevent accidental usage of a partially filled disk. Therefore, if the copying process is stopped before it is completed, the data module area is unusable. You can restart the copying process by reloading the copy/dump program, or you can restore the disk by reinitializing.

After successfully copying a disk, the copy program prints the message:

COPYPACK IS COMPLETE

Note: If you copy a disk containing an active checkpoint, that checkpoint exists on both the FROM and TO disks. When one of the two active checkpoints is used to restart the checkpointed program, care must be taken to avoid restarting the job a second time. To ensure that this will not occur, you can perform IPL and load Restart ($$RSTR) from the simulation area containing the second active checkpoint. If you then select the controlled cancel option when the H0nn halt occurs (nn is the last requested checkpoint number), the checkpoint is deactivated.

OUTPUT Parameters (COPYFILE)

The OUTPUT parameter is used for copying and printing card, tape, diskette or disk data files. It indicates whether you want the program to copy, print, or copy and print a file. The OUTPTX parameter can be used to display printed output in hexadecimal values. Definitions of the various OUTPUT parameters follow:

OUTPUT-DISK OUTPUT-FILE	Copy the file to disk or tape.
OUTPUT-PRINT	Print the file.
OUTPUT-BOTH	Copy the file to disk or tape, and print the file.
OUTPUT-MFCU OUTPUT-MFCU1 OUTPUT-MFCU2 OUTPUT-1442 OUTPUT-3741	Copy the file to the device named.
OUTPUT-'PRINT,MFCU' OUTPUT-'PRINT,MFCU1' OUTPUT-'PRINT,MFCU2' OUTPUT-'PRINT,1442' OUTPUT-'PRINT,3741'	Copy the file to the device named, and print the file.

INPUT Parameter (COPYFILE)

The INPUT parameter is used for copying from either the 3741 or a card device. INPUT-MFCU, INPUT-MFCU1, INPUT-MFCU2, INPUT-1442, and INPUT-3741 indicate that the input file is on the device named in the keyword parameter.

LENGTH Parameter (COPYFILE)

This parameter identifies the record length for the 3741 and is any number from 1 to 128. This keyword is optional whether the 3741 is being used as input or output. If this parameter is not specified, the record length defaults to 96.

When the 3741 is used, the LENGTH parameter must be equal to the record length in the HDR1 label on the 3741 and is any number from 1 to 128.

When the 3741 is used as output and the input is disk, card, or tape, the LENGTH parameter can be any number from 1 to 128 regardless of the record length of the disk, card or tape file being copied. If the record length specified on the 3741 is greater than the record length from the input file, the remainder of the record is filled with blanks (X'40'). If the record length from the disk, card, or tape file is greater than the LENGTH specified, the record is truncated.

This keyword is ignored if used with a device other than a 3741.

Copying Files

The copy/dump program can copy a file from disk, tape, cards or diskette to disk, tape, cards or diskette or from one area to another on the same disk.

The OCL load sequence for the copy/dump program indicates (1) the name and location of the disk or tape file being copied, and (2) the name and location of the disk or tape file being created. (See *OCL Considerations* in this section.)

In copying a file, the program can omit records. (See the description of the DELETE parameter for more information.)

In copying an indexed file, the program can reorganize records in the data portion in the order their keys appear in the index. (See the description of the REORG parameter for more information.)

Printing Files

The program can print all or part of a data file. To print only part, the program needs a SELECT control statement. (See the description of the SELECT control statement parameters in this section.) If you do not use a SELECT statement, the entire file is printed.

If you use SELECT KEY (PKY) or REORG-YES, records from indexed files are printed in the order their keys appear in the index portion of the file; otherwise, they are printed as they appear in the file. For each record, the program prints the record key followed by the contents of the record.

Records from sequential and direct files are printed in the order they appear in the file. For each record, the program prints the relative record number followed by the contents of the record.

The program uses as many lines as it needs to print the contents of a record. Appendix A lists the hexadecimal representation for characters in the standard character set.

The following example shows how the program prints hexadecimal numbers using OUTPTX:

```
ABCDE GHIJ12345
CCCCCBCCCDFFFFF4444444
1234567891123450000000
```

The hexadecimal number B6 represents a character that has no print symbol.

After printing the last record, the printer triple spaces and prints the following message:

(number) RECORDS PRINTED

DELETE Parameter (COPYFILE)

In copying a data file, the copy/dump program can omit records of one type. The DELETE parameter identifies the type of record. Use of the DELETE parameter is optional; if you do not use it, no records are deleted. DELETE cannot be used with direct files.

The form of the parameter is DELETE-'position, character'. *Position* is the position of the character in the records. *Character* is the character, except for apostrophes, blanks, or commas, that identifies the record. For example, with the parameter DELETE-'100,R' all records with an R in position 100 are deleted. By specifying the hexadecimal code for the character, you can use any character (including apostrophes, blanks, commas, and packed data) to identify the records to be deleted. For example, with the parameter DELETE-'100,X40', all records with a blank (hexadecimal 40) in position 100 are deleted.

Deleted records are always printed. If you are both copying and printing a data file, deleted records are printed with the other records. The deleted records are preceded by the word DELETED.

The OMIT keyword can be used instead of DELETE. The deleted records are not printed if OMIT is used.

REORG (Reorganize) Parameter (COPYFILE)

In copying an indexed file, the program can reorganize the file, so that the records in the data portion are in the same order as their keys in the file index. The REORG parameter tells the program whether or not to reorganize the file.

REORG-YES means reorganization; REORG-NO means no reorganization. REORG-NO is assumed if you omit the parameter.

If you tell the program to reorganize the file, the reorganization applies to the copy of the file rather than the original file. The original file is not affected.

Reorganization (REORG-YES) is required when you are both copying and printing an indexed file (OUTPUT-BOTH).

WORK Parameter (COPYFILE)

The WORK parameter applies to copying a data file from a data module mounted on drive 2 to another data module mounted on drive 2. It tells the program to use a work area on simulation area R1 on drive 1.

The parameter WORK-YES means that a work area is to be used. WORK-NO means no work area is used. WORK-NO is assumed if you omit the WORK parameter.

When you are copying on drive 2, the work area on R1 must contain a minimum of 198 contiguous unused tracks. If possible, R1 should not contain files or libraries because the number of data module changes on drive 2 decreases as R1 work space increases.

In copying the file, the program fills the work area with records from the file you are copying. Then it prints a message telling the operator to mount the other data module (the one to contain the copy) on drive 2. After transferring the records from the work area to the data module, the program prints another message telling the operator to remount the data module containing the file you are copying. The program repeats this procedure until all records have been transferred.

When WORK-YES is used, the input and output files must have different data module names. It is good practice to have different data module names on all data modules in an installation.

SELECT KEY and SELECT PKY Parameters (SELECT)

The SELECT KEY and SELECT PKY parameters apply to selecting part of an indexed file. The SELECT PKY parameter applies to selecting part of an indexed file that contains packed keys. The parameters are FROM and TO.

The FROM parameter (FROM-'key') gives the key of the first record to be selected. The TO parameter (TO-'key') gives the key of the last record to be selected. The record keys between those two in the file index identify the remaining records to be selected. If you want to select only one record, use the same record key in both the FROM and TO parameters.

For example, the parameters FROM-'000100' and TO-'000199' mean that records identified by keys 000100 through 000199 are to be selected.

If the file index does not contain the key you indicate in a FROM parameter, the program uses the next higher key in the index.

You can omit the TO parameter. If you do, the program assumes that the last key in the index is the TO key.

You can use fewer characters in the FROM or TO parameter than are contained in the actual keys; when keys are packed, however, you must use the same number of characters as contained in the actual keys. If you use fewer characters, the program ignores the remaining characters in the record key. The number of characters used in the FROM and TO parameters need not be the same.

For example, assume that the following are consecutive record keys in an index: A1000, A1119, A1275, A1900, A A1995, A2075, and 99999. The parameters FROM-'A1' and TO-'A199' refer to record keys A1000 through A1995.

If none of the keys in the file index begins with the characters you indicate in a FROM parameter, the program uses the key beginning with the next higher characters in the FROM parameter.

For example, assume that four consecutive record keys in an index begin with these characters: A1, A2, A8, and B1. The parameters FROM-'A3' and TO-'A9' would refer to keys beginning with the characters A8.

SELECT RECORD Parameters (SELECT)

The SELECT RECORD parameters can apply to any file, but are normally used for sequential and direct files. These parameters use relative record numbers to identify the records to be selected.

Relative record numbers identify a record's location with respect to other records in the file. The relative record number of the first record is 1, the number of the second record is 2, and so on.

The SELECT RECORD parameters are FROM and TO. The FROM parameter (FROM-number) gives the relative record number of the first record to be selected. The TO parameter (TO-number) gives the number of the last record to be selected. Records between those two records in the file are also selected.

For example, the parameters FROM-1 and TO-30 mean that the first 30 records (1-30) in the file will be selected.

You can omit the TO parameter. If you do, the program assumes that the number of the last record in the file is the TO number. If you want to select only one record, use the same number in the FROM and TO parameters.

FILE Parameter (SELECT)

This parameter allows only selected records to be copied to a disk, tape, cards, diskette, or printer.

LENGTH and LOCATION Parameters (KEY)

The KEY statement is used when building an indexed file from a sequential file. The LENGTH parameter specifies the length (1-29) of the key field. The LOCATION parameter specifies the starting location (1-65525) of the key field in the input record. When the KEY statement is used, the file described in the COPYO file statement must be a disk file and OUTPUT-DISK, OUTPUT-FILE, or OUTPUT-BOTH must be specified in the COPYFILE control statement.

CYLINDER Parameter (ACCESS)

This parameter identifies the cylinder number for start of file; the number can be between 0 and 202. For 5444, the number is the quotient obtained from dividing the file location by 2. For 5445, the number is indicated by the file location.

SECTOR Parameter (ACCESS)

This parameter is used to specify the sector on which the data to be copied is located. For a simulation area, it can be 0-47. For main data areas, it can be 1-48.

TRACK Parameter (ACCESS)

This parameter is used for 5445. The value can range from 0-19 and is specified by the file location.

RECL Parameter (ACCESS)

This parameter identifies the record length of the data in the file to be recovered. It can be 1-65536.

FROM Parameter (ACCESS)

This parameter identifies the unit on which the input data is located.

DISP Parameter (ACCESS)

This parameter specifies the displacement, in bytes, from the start of a sector to the beginning of a record in that sector. The number can be between 0 and 255.

COPYING MULTIVOLUME FILES

When multivolume files are copied, the first volume of the input file has to be online when the job is initiated. The output file must be a new file. If either condition is not satisfied, a halt occurs.

Maintaining Correct Date and Volume Sequence Numbers

To maintain the correct data and volume sequence numbers you must:

- Copy all the volumes of the file in one execution of $COPY, or
- Copy only one volume of the file in each execution of $COPY.

For example, if you copy a 3-volume file one volume at a time (volume 1 in the first execution, volume 2 in the second execution, and volume 3 in the third execution), the output file volumes will retain the original input date and volume sequence numbers. Or, if you copy all the volumes (1, 2, and 3) in the same execution, the system will assign the current system date and new volume sequence numbers in the output file. However, if you copy only volumes 2 and 3 in one execution, the output file volumes will be assigned the current system date and volume sequence numbers 1 and 2.

Maintaining Correct Relative Record Numbers

To maintain correct relative record numbers when copying one volume of a multivolume direct file, you must keep the output volume and the input volume equal in size. (If you want to increase the size of a file, you must copy the entire file.) If you copy the first volume of a 2-volume file and increase the number of records on that volume, you are also increasing relative record numbers of all the records on the next volume. Therefore, to maintain the correct relative record numbers, output and input volume extents (records or tracks) must be equal if you are copying only one volume of a multivolume direct file.

Direct File Attributes

If you copy an entire multivolume direct file in one run, the output file is given sequential attributes in the volume table of contents (VTOC). However, this does not affect file processing. A file with either sequential or direct attributes can be accessed by a consecutive or random access method. If only one volume is copied, the direct attribute is maintained.

Copying Multivolume Indexed Files

If you want to copy an indexed multivolume file, REORG-YES must be given in the COPYFILE statement. Since an unordered load to a multivolume indexed file is not permitted, a REORG-NO causes a halt. If you prefer not to reorganize the file, it must be copied one volume at a time. When you copy one volume at a time, the HIKEY on the output volume must be the same as the HIKEY on the input volume. If they are not equal, a halt occurs. Making the HIKEYs the same ensures that both the input and output volumes are the same length and no records will be lost. When you copy one volume of a multivolume indexed file, either REORG-YES or REORG-NO may be specified.

CARD AND DISKETTE CONSIDERATIONS ($COPY)

Card or Diskette Input

For card or diskette input files, end of file is determined by the presence of a record with /* in positions 1 and 2, and positions 3 through 80, 3 through 96 or 3 through 128 blank. This allows a card or diskette input file to contain /* records, assuming that at least one character is in columns 3 through 80, 3 through 96, or 3 through 128. A /& is handled the same as a /* record unless the input device is the system READER. The presence of a record with /& in positions 1 and 2 from the system READER is regarded as absolute end of file.

Card or Diskette Output

If the input record size (in bytes) is greater than the size of the card or diskette record, the input record is truncated. If the input record size is less than the size of the card or diskette record, the remaining portion of the card or diskette record contains blanks. For example, if the input file contains 60 byte records, the card is blank in columns 61 through 80 or 61 through 96. The diskette is blank in the remaining portion of the record length specified.

TAPE FILE CONSIDERATIONS

When copying or printing tape data files, you must describe the tape file being copied or printed and describe the file being created. The various tape record formats and labels are supported. (See *FILE Statement (Tape)* in Part 1 of this manual.) The tape file can be ASCII or EBCDIC. Default for record format (RECFM) is fixed length. On an unlabeled tape, record length (RECL) and block length (BLKL) must be specified.

Be careful when you copy a tape file with variable length records to disk or tape. The resulting file contains fixed-length records with a record length equal to the longest record length of the file copied from. Records copied with short record lengths have invalid information in the unused portion of the output record.

OCL CONSIDERATIONS

The following OCL statements are needed to load the copy/dump program if you are using the program to copy an entire disk:

```
// LOAD $COPY,code
// RUN
```

The code you supply depends on the location of the simulation area containing the copy/dump program. The codes are R1, F1, R2, and F2.

The following OCL statements are needed to do COPYFILE functions for disk and tape:

```
// LOAD $COPY,code
// FILE NAME-COPYIN,      (Required statement for
   parameters               input from disk or tape)

// FILE NAME-COPYO,       (Required statement for
   parameters               output to disk or tape)

// RUN
```

For information on the FILE statement parameters, see *OCL Statements* in Part 1 of this manual.

The UNIT parameter is required on each entered FILE statement. The allowable UNIT codes are R1, F1, R2, F2, D1, D2, T1, T2, T3, and T4 for COPYIN and COPYO file statements.

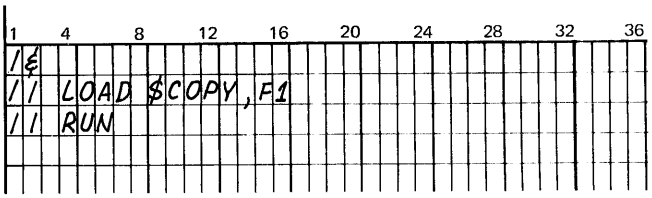
A FILE OCL statement is not required for a card, diskette, or printer file. The INPUT or OUTPUT keyword in the COPYFILE control statement must be used.

EXAMPLES

Figures 24 through 29 are examples of the OCL statements and control statements needed to (1) copy an entire disk, (2) copy a file from one disk to another, and (3) print part of a file.

Figures 30 through 41.1 are examples of the OCL statements and control statements needed to:

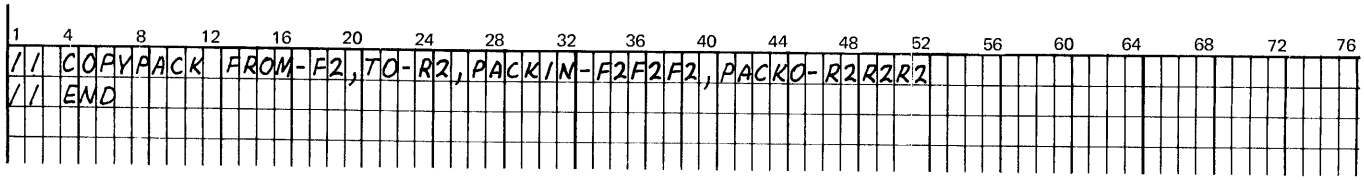
1. Copy a file from disk to tape.
2. Copy a file from tape to disk, printing part of the file.
3. Copy a file from tape to tape, selecting records to be copied.
4. Copy a card file to tape.
5. Copy a disk file to cards.
6. Copy a disk file to diskette.
7. Copy a tape file to diskette, printing part of the file.
8. Copy and print a portion of a file from diskette to disk.
9. Copy a card file to a diskette, printing the entire file.
10. Copy and print a portion of a file from diskette to cards.
11. Copy a card file to another card file.
12. Recover data from a main data area.



Explanation:

The copy/dump program is loaded from simulation area F1 on drive 1.

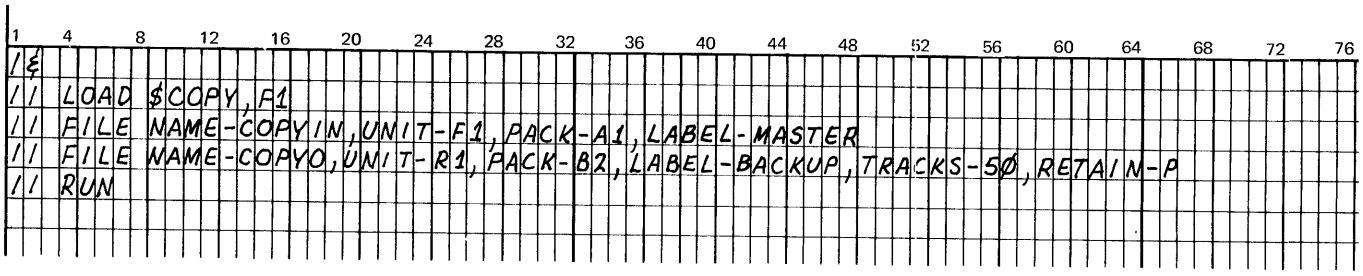
Figure 24. OCL Load Sequence for Copying an Entire Disk



Explanation:

The COPYPACK statement copies the contents of simulation area F2 (FROM-F2) with volume identification F2F2F2 (PACKIN-F2F2F2) onto simulation area R2 (TO-R2) with volume identification R2R2R2 (PACKO-R2R2R2).

Figure 25. Control Statements for Copying an Entire Disk



Explanation:

- Copy/dump program is loaded from simulation area F1 on drive 1.
- Input file (OCL sequence):
  1. Name that identifies file on disk is MASTER (LABEL-MASTER).
  2. Disk that contains the file is simulation area F1 on drive 1 (UNIT-F1). Its name is A1 (PACK-A1).
- Output file (OCL sequence):
  1. Name to be written on disk to identify the file is BACKUP (LABEL-BACKUP).
  2. Disk that is to contain the file is the simulation area R1 on drive 1 (UNIT-R1). Its name is B2 (PACK-B2).
  3. The file is to be permanent (RETAIN-P).
  4. The size of the file is 50 tracks (TRACKS-50).

Figure 26. OCL Load Sequence for Copying a File from One Disk to Another

```
// COPYFILE OUTPUT-DISK
// END
```

Explanation:

The COPYFILE statement tells the program to create the output file using all the data from the input file. The output file using all the data from the input file. The output file is a copy of the input file.

Figure 27. Control Statements for Copying a File from One Disk to Another

```
/&
// LOAD $COPY,F1
// FILE NAME-COPYIN,UNIT-D1,PACK-B2,LABEL-BACKUP
// RUN
```

Explanation:

- Copy/dump program is loaded from simulation area F1 on drive 1.
- Input file (OCL sequence):
  1. Name that identifies the file on disk is BACKUP (LABEL-BACKUP).
  2. Disk that contains the file is the main data area on drive 1 (UNIT-D1). Its name is B2 (PACK-B2).

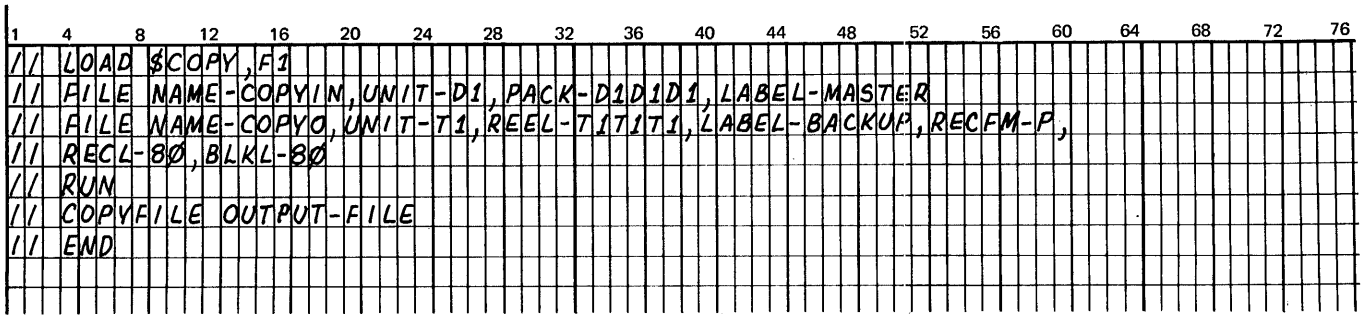
Figure 28. OCL Load Sequence for Printing Part of a File

```
// COPYFILE OUTPUT-PRINT
// SELECT KEY,FROM-'ADAMS',TO-'BAKER'
// END
```

Explanation:

- The file is being printed (COPYFILE statement).
- The file is an indexed file. The part being printed is identified by the record keys from ADAMS to BAKER in the index (SELECT statement).

Figure 29. Control Statements for Printing Part of a File

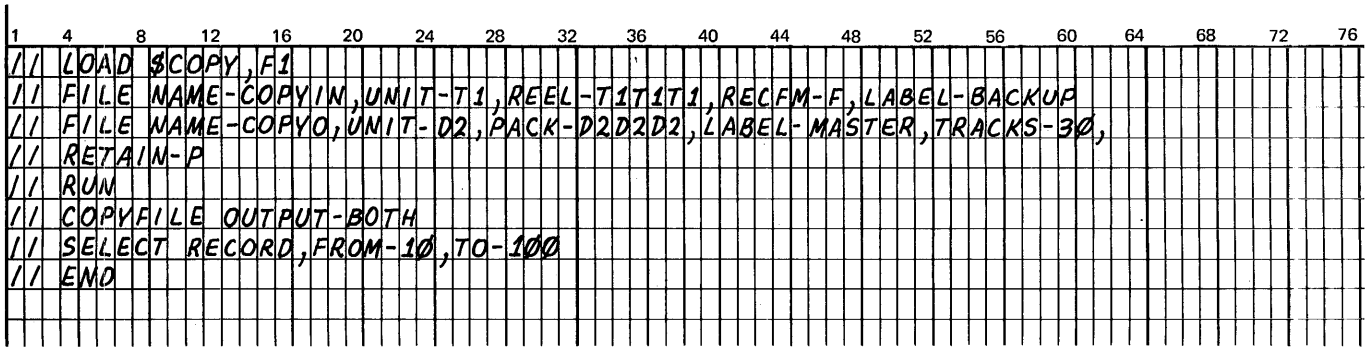


Explanation:

- Copy/dump program is loaded from simulation area F1 on drive 1.
- Input file (OCL sequence):
  1. Name that identifies the file on disk is MASTER (LABEL-MASTER).
  2. Disk that contains the file is main data area on drive 1 (UNIT-D1). Its name is D1D1D1 (PACK-D1D1D1).
- Output file (OCL sequence):
  1. Name to be written on tape to identify the file is BACKUP (LABEL-BACKUP).
  2. Tape unit that is to contain the file is tape unit 1 (UNIT-T1). Its name is T1T1T1 (REEL-T1T1T1).
  3. Record format used is fixed length, unblocked records (RECFM-F). The record length is 80 (RECL-80).
- Control statement explanation:

  The entire disk file named MASTER is copied to tape unit 1 (OUTPUT-FILE).

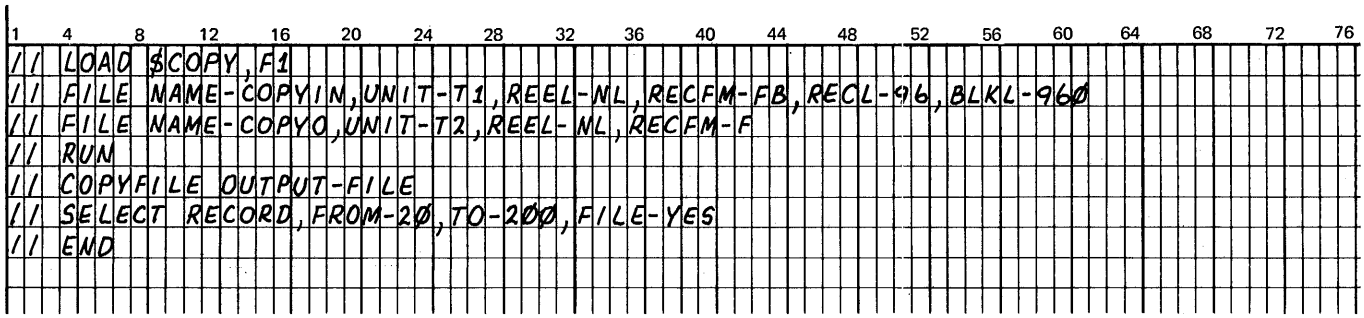
Figure 30. OCL and Control Statements to Copy a Disk File to a Tape File



Explanation:

- Copy/dump program is loaded from simulation area F1 on drive 1.
- Input file (OCL sequence):
  1. Name that identifies the file on tape is BACKUP (LABEL-BACKUP).
  2. Tape that contains the file is tape unit 1 (UNIT-T1). Its name is T1T1T1 (REEL-T1T1T1).
  3. Record format of the file is fixed length, unblocked records (RECFM-F).
- Output file (OCL sequence):
  1. Name to be written on disk to identify the file is MASTER (LABEL-MASTER).
  2. Disk that is to contain the file is the main data area on drive 2 (UNIT-D2). Its name is D2D2D2 (PACK-D2D2D2).
  3. The file is to be permanent (RETAIN-P).
  4. The size of the file is 30 tracks (TRACKS-30).
- Control statement explanation:
  1. The entire file is copied from tape to disk (OUTPUT-BOTH).
  2. Records 10 through 100 are printed (RECORD, FROM-10, TO-100).

Figure 31. OCL and Control Statements to Copy a Tape File to a Disk File and Print a Part of the File



Explanation:

- Copy/dump program is loaded from simulation area F1 on drive 1.
- Input file (OCL sequence):
  1. Tape that contains the file is tape unit 1 (UNIT-T1).
  2. Tape being copied is an unlabeled tape (REEL-NL); therefore, record format (RECFM-FB), record length (RECL-96), and block length (BLKL-960) are specified.
- Output file (OCL sequence):
  1. Tape unit that is to contain the file is tape unit 2 (UNIT-T2).
  2. No label is used on the output tape (REEL-NL).
  3. The record format is fixed length, unblocked (RECFM-F).
- Control statement explanation:
  1. Records 20 to 200 are copied (FILE-YES).
  2. No records are printed (OUTPUT-FILE).

Figure 32. OCL and Control Statements to Copy a Tape File to a Tape File and Select Records to be Copied

```
// LOAD $COPY,F1
// FILE NAME-COPYO,UNIT-T1,REEL-T1T1T1,LABEL-BACKUP,RECFM-FB
// RECL-96,BLKL-960
// RUN
// COPYFILE OUTPUT-FILE,INPUT-MFCU1
// END
```

Explanation:

- Copy/dump program is loaded from simulation area F1 on drive 1.
- Output file (OCL sequence):
  1. Name to be written on tape to identify the file is BACKUP (LABEL-BACKUP).
  2. Tape unit that is to contain the file is tape unit 1 (UNIT-T1). Its name is T1T1T1 (REEL-T1T1T1).
  3. Record format used is fixed length, blocked records (RECFM-FB). The record length is 96 (RECL-96); the block length is 960 (BLKL-960).
- Control statement explanation:

  The entire card file from the MFCU1 (INPUT-MFCU1) is copied to tape unit 1 (OUTPUT-FILE).

Figure 33. OCL and Control Statements to Copy a Card File to a Tape File

```
// LOAD $COPY,F1
// FILE NAME-COPYIN,UNIT-D1,PACK-D1D1D1,LABEL-MASTER
// RUN
// COPYFILE OUTPUT-1442
// END
```

Explanation:

- Copy/dump program is loaded from simulation area F1 on drive 1.
- Input file (OCL sequence):
  1. Name that identifies the file on disk is MASTER (LABEL-MASTER).
  2. Disk that contains the file is the main data area on drive 1 (UNIT-D1). Its name is D1D1D1 (PACK-D1D1D1).
- Control statement explanation:

  The entire disk file named MASTER is punched into cards by the 1442 (OUTPUT-1442).

Figure 34. OCL and Control Statements to Copy a Disk File to a Card File

```
// LOAD $COPY,F1
// FILE NAME-COPYIN,UNIT-D1,PACK-D1D1D1,LABEL-MASTER
// RUN
// COPYFILE OUTPUT-3741,LENGTH-128
// END
```

Explanation:

- Copy/dump program is loaded from simulation area F1 on drive 1.
- Input file (OCL sequence):
  1. Name that identifies the file on disk is MASTER (LABEL-MASTER).
  2. Disk that contains the file is main data area on drive 1 (UNIT-D1). Its name is D1D1D1 (PACK-D1D1D1).
- Control statement explanation:

  The entire disk file named MASTER is copied to the 3741 (OUTPUT-3741). The record length of the file on the 3741 is 128 (LENGTH-128).

Figure 35. OCL and Control Statements to Copy a Disk File to the 3741

```
// LOAD $COPY,F1
// FILE NAME-COPYIN,UNIT-T1,REEL-PAYROL,RECFM-F,LABEL-MASTER
// RUN
// COPYFILE OUTPUT-'PRINT,3741',LENGTH-96
// SELECT RECORD,FROM-4,TO-120
// END
```

Explanation:

- Copy/dump program is loaded from simulation area F1 on drive 1.
- Input file (OCL sequence):
  1. Name that identifies the file on tape is MASTER (LABEL-MASTER).
  2. Tape that contains the file is tape unit 1 (UNIT-T1). Its name is PAYROL (REEL-PAYROL).
  3. Record format of the file is fixed length, unblocked records (RECFM-F).
- Control statements explanation:
  1. The entire file is copied from tape to the 3741 (OUTPUT-'PRINT,3741'). The record length of the file on the 3741 is 96 (LENGTH-96).
  2. Records 4 through 120 are printed (RECORD,FROM-4,TO-120).

Figure 36. OCL and Control Statements to Copy a Tape File to a Diskette File and Print a Part of the File

```
// LOAD $COPY,F1
// FILE NAME-COPY0,UNIT-D2,PACK-D2D2D2,LABEL-SALES,TRACKS-15,RETAIN-T
// RUN
// COPYFILE OUTPUT-BOTH,INPUT-3741,LENGTH-50
// SELECT RECORD,FROM-5,TO-250,FILE-YES
// END
```

Explanation:

- Copy/dump program is loaded from simulation area F1 on drive 1.
- OUTPUT file (OCL sequence):
  1. Name to be written on disk is SALES (LABEL-SALES).
  2. Disk that is to contain the file is main data area on drive 2 (UNIT-D2). Its name is D2D2D2 (PACK-D2D2D2).
  3. The file is to be temporary (RETAIN-T).
  4. The size of the file is 15 tracks (TRACKS-15).
- Control statements explanation:
  1. Records 5 to 250 are copied (FILE-YES) and printed (OUTPUT-BOTH).
  2. Input is the 3741 (INPUT-3741) and the record length in the HDR1 label on the 3741 is 50 (LENGTH-50).

Figure 37. OCL and Control Statements to Copy a Diskette File to a Disk File and Print Only the Copied Records

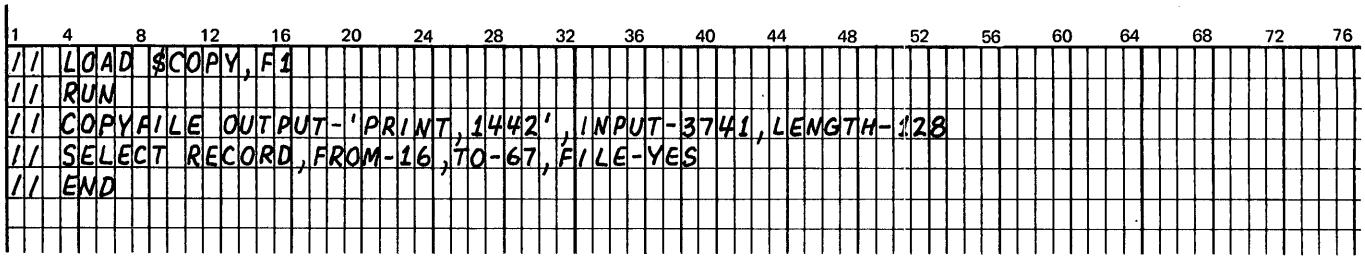
```
// LOAD $COPY,F1
// RUN
// COPYFILE OUTPUT-'PRINT,3741',INPUT-MFCU1,LENGTH-96
// END
```

Explanation:

- Copy/dump program is loaded from simulation area F1 on drive 1.
- Control statement explanation:

  The entire card file from the MFCU1 (INPUT-MFCU1) is copied to the 3741 and printed (OUTPUT-'PRINT,3741'). The record length of the output file on the 3741 is 96 (LENGTH-96).

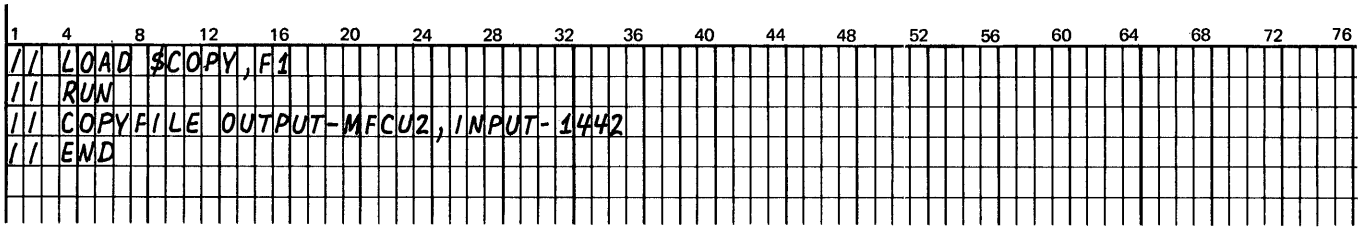
Figure 38. Control Statement to Copy a Card File to a Diskette and Print the Entire File



Explanation:

- Copy/dump program is loaded from simulation area F1 on drive 1.
- Control statement explanation:
  1. Records 16 through 67 are copied (FILE-YES) to the 1442 and printed (OUTPUT-'PRINT,1442').
  2. Input is the 3741 (INPUT-3741), and the record length in the HDR1 label on the 3741 is 128 (LENGTH-128).

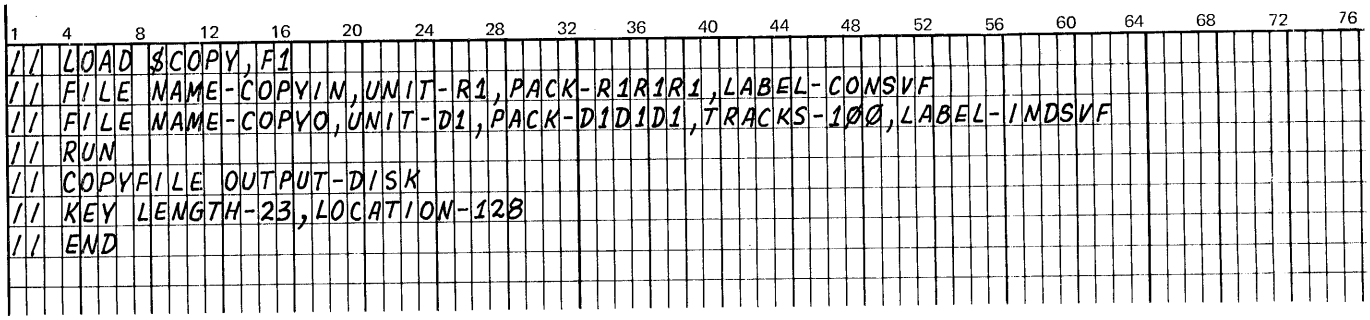
Figure 39. Control Statements to Copy and Print a Portion of a File on a Diskette to a Card Device



Explanation:

- Copy/dump program is loaded from simulation area F1 on drive 1.
- Control statement explanation:
  1. The entire file is copied to the MFCU2 (OUTPUT-MFCU2).
  2. Input is the 1442 (INPUT-1442).

Figure 40. Control Statement to Copy a Card File to Another Card File



Explanation:

- Copy/dump program is loaded from simulation area F1 on drive 1.
- Input file (OCL sequence):
  1. Name that identifies file on simulation area is CONSVF (LABEL-CONSVF).
  2. Disk that contains the file is the simulation area R1 on drive 1 (UNIT-R1). Its name is R1R1R1 (PACK-R1R1R1).
- Output file (OCL sequence):
  1. Name to be written on main data area to identify the file is INDSVF (LABEL-INDSVF).
  2. Disk that is to contain the file is the main data area on drive 1 (UNIT-D1). Its name is D1D1D1 (PACK-D1D1D1).
  3. The size of the file is 100 tracks (TRACKS-100).
- The COPYFILE statement tells the program to create the output file using all the data from the input file.
- The KEY statement tells the program to create an index for the output file consisting of 23-byte keys (LENGTH-23) which are located 128 bytes into the record (LOCATION-128).

Figure 41. Control Statements to Copy a Sequential File From a Simulation Area to a Main Data Area and Create an Indexed Output File

```
// LOAD $COPY,F1
// FILE NAME-COPY0,UNIT-D2,PACK-D2D2D2,TRACKS-100,RETAIN-LABEL-MASTER
// RUN
// COPYFILE OUTPUT-DISK
// ACCESS FROM-D1,CYLINDER-159,TRACK-Ø,SECTOR-1,DISP-Ø,RECL-256
// SELECT RECORD,FROM-1,TO-1349,FILE-YES
// END
```

Explanation:

- Copy/dump program is loaded from simulation area F1 on drive 1.
- Output file (OCL sequence):
  1. Name to be written on main data area to identify the file is MASTER (LABEL-MASTER).
  2. Disk that is to contain the file is the main data area on drive 2 (UNIT-D2). Its name is D2D2D2 (PACK-D2D2D2).
  3. The size of the file is 100 tracks (TRACKS-100).
- The COPYFILE statement tells the program to create the output file using all the data from the input file.
- The ACCESS statement identifies the location of the data to be copied as being on D1 (FROM-D1), at cylinder, track, sector, displacement 159/0/1/0 (CYLINDER-159,TRACK-0,SECTOR-1,DISP-0), and that the records are 256 bytes long (RECL-256).

Note: // SELECT RECORD with FILE-YES must be specified when using // ACCESS.

Figure 41.1. Control Statements to Recover Data From Main Data Area D1

Simulation Area Program—$SCOPY

The simulation area program has the following six functions:

- COPYAREA: Copies the entire contents of one simulation area or simulation backup area to another simulation area or simulation backup area.
- CLEAR: Clears all the data from a simulation area or simulation backup area and builds a simulated cylinder 0 (optionally gives volume ID and owner ID).
- NEWNAME: Changes the name (volume ID) of a simulation area or simulation backup area.
- NAMES: Prints the name (volume ID) of each available simulation area, simulation backup area, and main data area.
- MOVE: Copies the entire contents of one simulation area or simulation backup area to another simulation area or simulation backup area, clears the area from which the contents were copied, and builds a simulated cylinder 0 in the area copied from.
- COPYIPL: Copies IPL records from one 3340 data module to another 3340 data module.

The use of any of these functions requires that the simulation area referenced be dedicated to program level executing $SCOPY. The data module, on which the simulation area is being referenced, cannot be dedicated to the other level.

Four contiguous areas of 10 cylinders each (starting at cylinder 169) are reserved on each of the 3340 data modules to simulate 5444 drives. The first two areas on D1 are reserved for F1 and R1; the first two areas on D2 are reserved for F2 and R2. These four areas are accessible via normal data management (except multivolume and indexed files) and Model 12 system utility programs except $ALT, $BUILD, $INIT, and $RSALT. $SCOPY provides access to simulation areas and simulation backup areas for maintenance purposes.

The simulation areas are designated as follows:

Area		Start (CCC/HH/RR)	End (CCC/HH/RR)
A	First simulation area	169/00/01	178/19/48
B	Second simulation area	179/00/01	188/19/48
C	First simulation backup area	189/00/01	198/19/48
D	Second simulation backup area	199/00/01	208/19/48

CONTROL STATEMENT SUMMARY

Function	Control Statements
COPYAREA	// COPYAREA FROM-code,TO-code,PACK-name,AREA-name[,TONAME-name] [SYSTEM-{YES / NO}]
CLEAR	// CLEAR FROM-code,PACK-name[,AREA-name] [,CLRNAME-name] [,ID-name] [TYPE- {CHECK / FORCE}]
NEWNAME	// NEWNAME TO-code,PACK-name,AREA-name,TONAME-name
NAMES	// NAMES [PRINT]
MOVE	// MOVE FROM-code,TO-code,PACK-name,AREA-name[,TONAME-name] [,ID-name] [,SYSTEM-{YES / NO}] [,CLRNAME-name]
COPYIPL	// COPYIPL FROM-D1,TO-D2,PACK-name
	// END

PARAMETER SUMMARY

COPYAREA

FROM-code	Identifies the data module and the simulation area or simulation backup area being copied. Possible codes are D1A, D1B, D1C, D1D, D2A, D2B, D2C, and D2D.
TO-code	Identifies the data module and the simulation area or simulation backup area receiving the copy (see FROM-code description for possible codes).
PACK-name	Identifies the name of the data module receiving the copy.
AREA-name	Identifies the name of the simulation area or simulation backup area being copied.
TONAME-name	Specifies a name change for the area receiving the copy.
SYSTEM- YES / NO	Specifies whether IPL information is to be copied.

CLEAR

FROM-code	Identifies the data module and simulation area or simulation backup area being cleared (see COPYAREA FROM-code for possible codes).
PACK-name	Specifies the data module containing the area to be cleared.
AREA-name	Specifies the area to be cleared. Cannot be specified if AREA has no assigned name. PID001 must be specified to clear an area used for distribution of programs from the IBM program library/PID. The name PID001 should only be used for this purpose.
CLRNAME-name	Specifies the name to be given to the area being cleared. If no parameter is specified, the name of the area is the name previously defined.
ID-name	Enables you to use a 10-character name in addition to the area name to further identify a disk.
TYPE-CHECK	Tells the program to check for active files or libraries and halt if any are found before clearing the area.
TYPE-FORCE	Tells the program to clear the area without checking for active files or libraries.

NEWNAME

TO-code	Specifies the name of the data module and the simulation area or simulation backup area being renamed. (See COPYAREA FROM-code for possible codes.)
PACK-name	Specifies the name of the data module containing the area being renamed.
AREA-name	Specifies the existing name of the simulation area or simulation backup area being renamed.
TONAME-name	Specifies the new name being given to the simulation area or simulation backup area.

NAMES

PRINT	Specifies that the names of all online simulation areas and simulation backup areas are to be printed on the system print device.

MOVE

FROM-code	Identifies the data module and the simulation area or simulation backup area being moved. Possible codes are D1A, D1B, D1C, D1D, D2A, D2B, D2C, and D2D.
TO-code	Identifies the data module and simulation area or simulation backup area receiving moved information (see FROM-code for possible codes).
PACK-name	Identifies the name of the data module containing the simulation area or simulation backup area receiving the moved information.
AREA-name	Specifies the name of the simulation area or simulation backup area being moved.
TONAME-name	Specifies a name change for the simulation area or simulation backup area receiving the moved information.
ID-name	Specifies the owner ID.
SYSTEM-YES / NO	Specifies whether IPL information is to be moved.
CLRNAME-name	Used to assign a name to the area from which the information has been moved.

COPYIPL

FROM-D1	Identifies the data module containing the IPL records to be copied.
TO-D2	Identifies the data module receiving the IPL records.
PACK-name	Identifies the name of the data module receiving the IPL records.

PARAMETER DESCRIPTIONS

FROM and TO Parameters (COPYAREA)

The FROM parameter (FROM-code) identifies the data module and the simulation area or simulation backup area that contains the information to be copied. The TO parameter (TO-code) identifies the data module and the simulation area or simulation backup area that is to receive the copy. Possible codes are D1A, D1B, D1C, D1D, D2A, D2B, D2C, and D2D.

PACK Parameter (COPYAREA)

The PACK parameter (PACK-name) identifies the name of the data module containing the simulation area or simulation backup area receiving the copy. This is the name assigned by the disk initialization program ($INIT).

AREA Parameter (COPYAREA)

The AREA parameter (AREA-name) identifies the name of the simulation area or simulation backup area that is to be copied.

Note: Using a COPYAREA or MOVE statement, the receiving area is assigned the owner ID of the area being copied from.

TONAME Parameter (COPYAREA)

The TONAME parameter (TONAME-name) is used to change the name of the simulation area or simulation backup area that is to receive the copy. The name may contain up to 6 characters (see *CLRNAME Parameter (CLEAR)* for explanation of valid names). If the TONAME parameter is omitted, the name of the simulation area or simulation backup area that is to be copied is used.

SYSTEM Parameter (COPYAREA)

The SYSTEM parameter is used to copy IPL information. If SYSTEM-YES is specified, the IPL information from cylinder 0 of the system data module on drive 1 is copied to cylinder 0 of the data module receiving the copied information. If SYSTEM-NO is specified, the IPL information is not copied. If no parameter is specified, SYSTEM-NO is assumed.

FROM Parameter (CLEAR)

The FROM parameter (FROM-code) identifies the data module and the simulation area or simulation backup area to be cleared. Codes that may be used are D1A, D1B, D1C, D1D, D2A, D2B, D2C, and D2D.

PACK Parameter (CLEAR)

The PACK parameter (PACK-name) specifies the name of the data module containing the simulation area or simulation backup area that is to be cleared. This is the name assigned by the disk initialization program ($INIT).

AREA Parameter (CLEAR)

The AREA parameter (AREA-name) specifies the name of the simulation area or simulation backup area that is to be cleared. This parameter cannot be specified if the area has no assigned name. The AREA parameter must be specified as PID001 in order to clear an area used for distribution of programs from the IBM program library/PID. The name PID001 should be used only for this purpose.

CLRNAME Parameter (CLEAR)

The CLRNAME parameter (CLRNAME-name) specifies the name to be given to the area that is to be cleared. The name may be up to 6 characters in length and contain any combination of standard System/3 characters except apostrophes, embedded blanks, and commas (due to their delimiter function). (See Appendix A for a list of standard System/3 characters.) Valid area names are 0, F0001, 012, A1B9, and ABC. If no parameter is specified, the name of the area is the name previously defined. If no name has been previously defined, CLRNAME must be specified.

ID Parameter (CLEAR)

The ID parameter (ID-name) enables you to include a maximum of 10 characters, in addition to the area name, to further identify a simulation area or simulation backup area. (See *CLRNAME Parameter (CLEAR)* for explanation of valid names.) The information is strictly for area identification. (It is not used by the system for checking purposes.) If no parameter is specified, the owner ID area in the volume label is left blank.

TYPE Parameter (CLEAR)

The TYPE parameter specifies the type of clear that is to be done. If TYPE-CHECK is specified, a check is made for active files or libraries. If any are found, the system halts. If TYPE-FORCE is specified, the area is cleared without a check for active files or libraries. (All libraries and data files are deleted.)

TO Parameter (NEWNAME)

The TO parameter (TO-code) identifies the data module and the simulation area or simulation backup area that is to be renamed. The possible codes are D1A, D1B, D1C, D1D, D2A, D2B, D2C, and D2D.

PACK Parameter (NEWNAME)

The PACK parameter (PACK-name) specifies the name of the data module containing the simulation area or simulation backup area being renamed. This is the name assigned by the disk initialization program ($INIT).

AREA Parameter (NEWNAME)

The AREA parameter (AREA-name) specifies the existing name of the simulation area or simulation backup area that is to be renamed.

TONAME Parameter (NEWNAME)

The TONAME parameter (TONAME-name) specifies the new name to be given to the simulation area or simulation backup area. The new name may be up to 6 characters in length. (See *CLRNAME Parameter (CLEAR)* for an explanation of valid names.)

PRINT Parameter (NAMES)

The PRINT parameter indicates that all online simulation area names or simulation backup area names are to be printed on the system print device. If no parameter is specified, the simulation area names or simulation backup area names are printed. If an area is unavailable or being used by the other program level, its volume ID is left blank and an exception line is printed, giving the reason.

FROM and TO Parameters (MOVE)

The FROM parameter (FROM-code) identifies the data module and the simulation area or simulation backup area that is to be moved. The TO parameter (TO-code) identifies the data module and simulation area or simulation backup area that is to receive the moved information. Possible codes are D1A, D1B, D1C, D1D, D2A, D2B, D2C, and D2D.

PACK Parameter (MOVE)

The PACK parameter (PACK-name) identifies the name of the data module containing the simulation area or simulation backup area that is to receive the moved information. The name was assigned by the disk initialization program ($INIT).

AREA Parameter (MOVE)

The AREA parameter (AREA-name) specifies the name of the simulation area or simulation backup area to be moved.

TONAME Parameter (MOVE)

The TONAME parameter (TONAME-name) is used to change the name of the simulation area or simulation backup area that is to receive the information. If no parameter is specified, the name of the simulation area or simulation backup area that is to be moved is used. (See *CLRNAME Parameter (CLEAR)* for an explanation of valid names.)

ID Parameter (MOVE)

The ID parameter (ID-name) specifies the owner ID that is to be given to the area from which information was moved. If no parameter is specified, the owner ID in the volume label is left blank.

Note: Using a COPYAREA or MOVE statement, the receiving area is assigned the owner ID of the area being copied from. The owner ID name may be up to 10 characters in length. (See *CLRNAME Parameter (CLEAR)* for an explanation of valid names.)

SYSTEM Parameter (MOVE)

The SYSTEM parameter is used to move IPL information. If SYSTEM-YES is specified, the IPL information from cylinder 0 of the system data module on drive 1 is moved to cylinder 0 of the data module receiving the moved information. If SYSTEM-NO is specified, the IPL information is not moved. If no parameter is specified, SYSTEM-NO is assumed.

CLRNAME Parameter (MOVE)

The CLRNAME parameter (CLRNAME-name) is used to assign a name to the area from which the information has been moved. The name may be up to 6 characters. (See *CLRNAME Parameter (CLEAR)* for an explanation of valid names.) If no parameter is specified, the area is cleared and the name previously assigned is used.

FROM and TO Parameter (COPYIPL)

The FROM parameter (FROM-D1) identifies the data module containing the IPL records that are to be copied. The TO parameter (TO-D2) identifies the data module that is to receive the IPL records.

Note: COPYIPL can only be from D1 to D2.

PACK Parameter (COPYIPL)

The PACK parameter (PACK-name) identifies the name of the data module that is to receive the IPL records. The name assigned by the disk initialization program ($INIT).

OCL CONSIDERATIONS

The following OCL statements are needed to load the simulation area program:

```
// LOAD $SCOPY,code

// RUN
```

The code you supply depends on the location of the simulation area containing the simulation area program. The codes are R1, F1, R2, and F2.

EXAMPLES

Figures 42 through 48 are examples of control statements used to perform specific functions of the simulation area program.

```
1   4   8   12  16  20  24  28  32  36  40  44  48  52  56  60  64  68  72  76
//  CLEAR FROM-D2C,PACK-D2D2D2,CLRNAME-D2CD2C,ID-BACKUPF1
```

Explanation:

After a check for active files and libraries (default is TYPE-CHECK), the first backup area on drive 2 is cleared. It is given a volume ID of D2CD2C and an owner ID of BACKUPF1. This is an example of the CLEAR that is to be run after the entire data module has been initialized by $INIT.

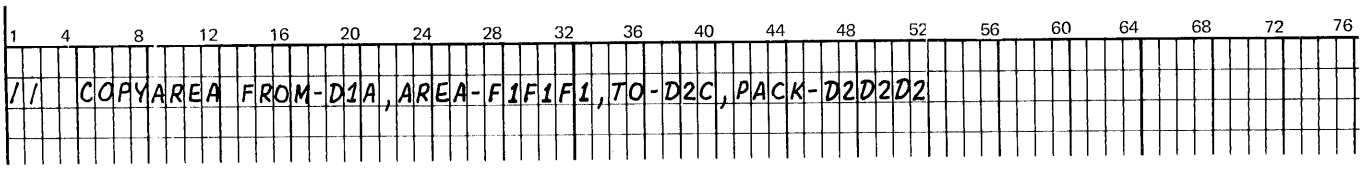
Figure 42. CLEAR Example: Clearing a Simulation Backup Area

```
1   4   8   12  16  20  24  28  32  36  40  44  48  52  56  60  64  68  72  76
//  CLEAR FROM-D1C,PACK-D1D1D1,AREA-PIDØØ1,CLRNAME-D1CD1C,TYPE-FORCE
```

Explanation:

After verification that the volume ID on the third area of drive D1 is PID001, the area is cleared and given a volume ID of D1CD1C. The owner ID is all blanks and the check for active files and libraries is bypassed. This is an example of the control statement needed to clear an area containing programs from the IBM program library/PID.

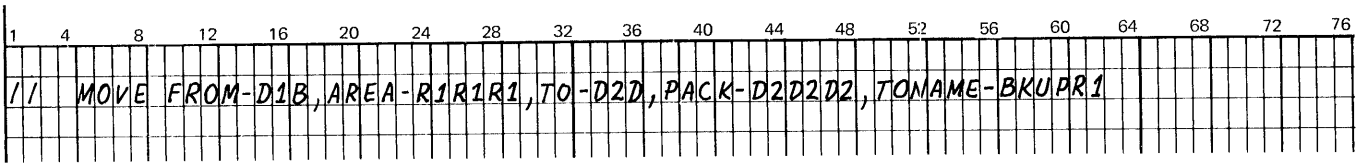
Figure 43. CLEAR Example: Clearing an Area Containing IBM Programs



Explanation:

After verification that the volume ID of area D1A is F1F1F1, the area (D1A) is copied to the first backup area on drive 2. The entire simulation area is copied including cylinder 0, the volume ID, and the owner ID if it was present on D1A.

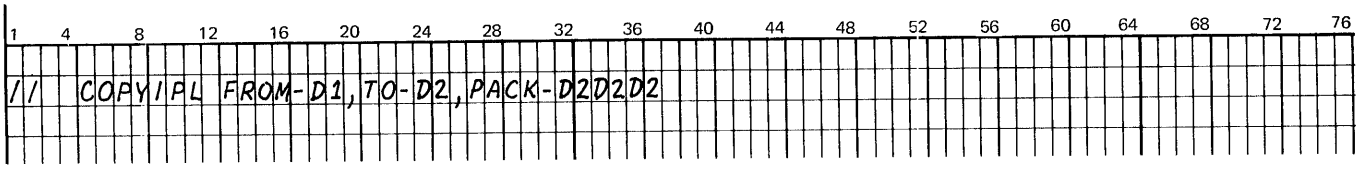
Figure 44. COPYAREA Example: Copy an Entire Simulation Area



Explanation:

The entire R1 simulation area on drive 1 is copied to the second backup area on drive 2 and the D2D area is given a volume ID of BKUPR1 and an owner ID of the R1 area if one exists. After the copy is complete, the R1 simulation area is cleared of all data, its owner ID field is blank, and it retains its volume ID of R1R1R1. The R1 simulation area is now ready to be the receiving area of a COPYAREA or another MOVE.

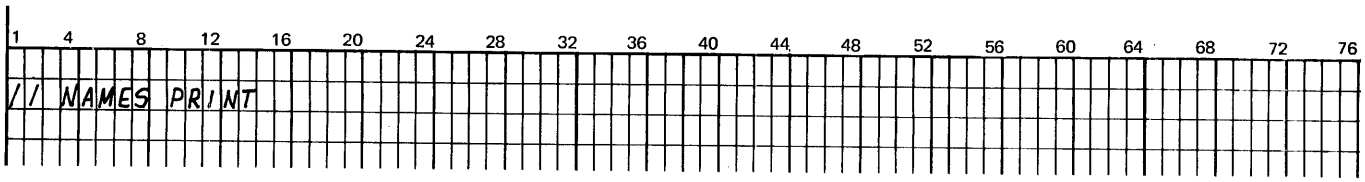
Figure 45. MOVE Example: Copy an Entire Simulation Area With New Volume ID



Explanation:

The IPL (initial program load) records and the 3340 microcode are copied from cylinder 0 of the data module on drive 1 to cylinder 0 of the data module on drive 2. A check is made before the copy to ensure that the volume ID of the data module on drive 2 is D2D2D2.

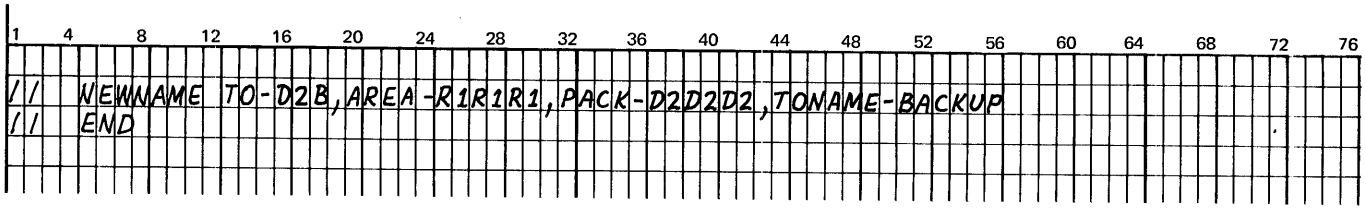
Figure 46. COPYIPL Example: Copy Cylinder 0 From Drive 1 to Drive 2



Explanation:

This control statement enables you to print on the system print device the volume ID of all online and available data modules and simulation and backup areas. All simulation and backup areas on a data module are considered by the simulation area program as unavailable if the data module is dedicated to the other program level, if the other program level has a rollin pending, or if the data module has not been initialized by System/3 $INIT. This control statement also provides the capability to print an exception line, if needed, giving the reason for any unavailable simulation or backup area.

Figure 47. PRINT Example: Print ID Information



Explanation:

After verifying that the name (volume ID) of the data module on D2 is D2D2D2 and that the name (volume ID) of the second backup area on D2 is R1R1R1, the program changes the name (volume ID) of the backup area from R1R1R1 to BACKUP.

Figure 48. NEWNAME Example: Changing Volume ID

The following diagram shows the location of data modules and backup areas on the 3340:

	Cyl 0	Cyl 1-166	Cyl 167-168	Cyl 169-178	Cyl 179-188	Cyl 189-198	Cyl 199-208	Cyl 209 heads 1-7
Drive 1	System area	Main data area D1	Alternate tracks	Simulation area A D1A F1 area	Simulation area B D1B R1 area	Simulation area C D1C	Simulation area D D1D	Error logging and CE tracks
	Cyl 0	Cyl 1-166	Cyl 167-168	Cyl 169-178	Cyl 179-188	Cyl 189-198	Cyl 199-208	Cyl 209 heads 1-7
Drive 2	System area	Main data area D2	Alternate tracks	Simulation area A D2A F2 area if simulation is on	Simulation area B D2B R2 area if simulation is on	Simulation area C D2C	Simulation area D D2D	Reserved

Library Maintenance Program—$MAINT

The library maintenance program has five functions:

Function	Meaning
Allocate	Create (reserve space for), delete, reorganize, and change the sizes of libraries; create the scheduler work area and rollout/rollin area on a system simulation area.
Copy	Place entries in and display the contents of libraries. Create a file from library entries.
Delete	Delete library entries.
Modify	Modify source library entries.
Rename	Change the names of library entries.

The control statements you must supply depend on the function you are using.

All simulation areas referenced by the control statements must remain online during the library maintenance run.

LIBRARY DESCRIPTION

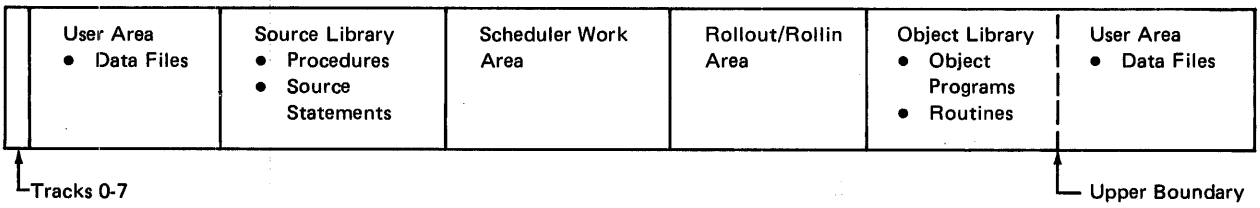
The *source library* is an area on disk for storing procedures and source statements. *Procedures* are groups of OCL statements used to load programs. The statements can be followed by input data for the programs. (Procedures for utility programs can, for example, contain utility control statements.) *Source statements* are sets of data, the most common of which are RPG II source programs and disk sort sequence specifications.

The *object library* is an area on disk for storing object programs and routines. *Object programs* are programs and subroutines in such a form that they can be loaded for execution. (They are sometimes called executable object programs.) *Routines* are programs and subroutines that need to be link-edited into object programs before they can be loaded for execution. (They are sometimes called non-executable object programs.)

Location of Libraries on Disk

Libraries cannot exist in the main data area; only R1, F1, R2, and F2 can contain libraries that may be referenced by the library maintenance program.

The location of a source library with respect to an object library is always the same:



The boundaries of a source library are fixed. They can be changed only by the allocate function of the library maintenance program. The upper boundary of an object library, however, can be moved as additional space is needed when entries are placed in the library. This happens only if space is available following the library and if the entries being placed beyond the normal boundary are *not* permanent entries.

Organization of Library Entries

Object Library

Entries are stored in the object library serially; that is, a 20-sector program occupies 20 consecutive sectors. Temporary entries follow all permanent entries in the object library.

If necessary, the upper boundary is changed to allow more space for temporary entries. The upper boundary of the library is extended to the end of the pack or to the first temporary or permanent file, allowing the maximum amount of space for the temporary library entry. At the successful completion of the copy, the upper boundary is returned to the track boundary at the end of the last temporary entry. If the copy was not completed successfully, the upper boundary may remain extended. When a permanent entry is placed in the library or the library is reorganized, all temporary entries are deleted and the upper boundary returns to its original location. Permanent entries cannot exceed the original upper boundary.

Gaps can occur in the object library when an entry is deleted. The associated directory entries point to these gaps. When the library maintenance program places a new entry in the library, it searches the directory for a gap that has the same number of sectors, or the fewest sectors over the number required by the new entry. If the entry is smaller than the gap, the last part of the gap is not pointed to by a directory entry. Since this gap has no directory entry, it cannot be used until the library is reorganized.

If the number of unusable sectors becomes excessive, the library should be reorganized. In reorganizing entries, the library maintenance program deletes temporary entries and shifts entries so that gaps do not appear between them. This makes more sectors available for use.

Source Library

The source library differs from the object library in that entries within the source library need not be stored in consecutive sectors. An entry can be stored in many widely separated sectors with each sector pointing to the sector that contains the next part of the entry. When an entry is placed in the source library, it is placed in as many sectors as required regardless of where the sectors are located within the library.

The boundary of the source library cannot be expanded; therefore, an entry must fit within the available library space. To provide as much space as possible within the prescribed limits of the source library, the system compresses entries. That is, all duplicate characters are removed from entries. Later, if the entries are printed or punched, the duplicate characters are reinserted.

When the size of the source library is changed or the source library is reorganized, all temporary entries are deleted.

Library Directories

The program creates a separate directory for each library. Every library entry has a corresponding entry in its library directory. The directory entry contains such information as the name and location of the library entry. The program also creates a system directory, which contains information about the size and available space in the libraries and their directories.

Organization of this Section

The five functions of the library maintenance program are described separately. Every description contains the following:

- List of specific uses
- Control statement summary indicating the form of control statement needed for each use
- Parameter descriptions explaining, in detail, the contents and meanings of the parameters
- Function descriptions explaining the details of each function

Following the function descriptions are:

- OCL considerations
- Examples

ALLOCATE FUNCTION

Uses

- Create (reserve space for) libraries, scheduler work area, and rollout/rollin area.
- Change the sizes of libraries.
- Delete libraries.
- Reorganize libraries.

Control Statement Summary

// ALLOCATE TO-code,SOURCE-{number / R},OBJECT-{number / R},SYSTEM-{NO / YES},DIRSIZE-number,WORK-code

	Use①	Parameter Needed②
Source Library	Create	TO-code,SOURCE-number,WORK-code③
	Change Size	TO-code,SOURCE-number,WORK-code
	Delete	TO-code,SOURCE-0
	Reorganize	TO-code,SOURCE-R,WORK-code
Object Library	Create	TO-code,OBJECT-number,SYSTEM-{NO / YES}
	Change Size	TO-code,OBJECT-number,WORK-code④
	Delete	TO-code,OBJECT-0
	Reorganize	TO-code,OBJECT-R,WORK-code④

① You can indicate a source library use, any object library use, or uses involving both libraries (for example, deleting the source library and changing the size of the object library).

② If you are indicating uses for both libraries, use only one TO parameter. (The libraries must be in the same simulation area.) Also, use only one WORK parameter if both uses require a WORK parameter.

③ The WORK parameter is needed only if the simulation area contains an object library that you are not deleting.

④ The WORK parameter is not required if this is a compress in place.

Considerations and Restrictions

This program has restrictions and operating conditions that the user must be aware of when maintaining libraries.

Allocation of Disk Space

The library maintenance program allocates disk space for each of the following functions:

- Creating a library.
- Increasing the size of a library.
- Reorganizing a library.
- Dynamically extending an object library to copy temporary entries to the library.
- Sorting a directory before it is printed.
- Modifying a source library entry.

The space allocated by the program is the first contiguous space large enough for the function to be performed. The library maintenance program uses as much space as is available to the end of the simulation area or to the first temporary or permanent data file, removing all scratch files in this area. If, within a single load of the program, there are functions performed requiring more than four disk areas to be allocated, a halt occurs. The library maintenance program must be reloaded to continue.

Removing Temporary Entries

When a library is reorganized, changed in size, or moved, all temporary entries in that library are deleted. This applies to both the source and object libraries.

Library Restrictions

The allocate function cannot reference the libraries on the simulation area from which the library maintenance program or the system was loaded. For example, if the system was loaded (IPL) from F1 and the library maintenance program was loaded from R1, the source or object libraries on F1 and R1 cannot be referenced on an ALLOCATE statement.

Moving the Object Library

When the user creates or changes the size of the source library in a simulation area that contains an object library, the object library is moved and reorganized, and all temporary entries are deleted.

Control Statement Restrictions

The SOURCE or OBJECT parameter must be specified on the ALLOCATE statement. If the SYSTEM or DIRSIZE parameter is specified, the OBJECT parameter must also be specified.

Procedure Restrictions

If nested procedures are used, information contained in the scheduler work area can become invalid when a source library is reorganized or changed in size. Therefore, if a procedure is used to reallocate or reorganize libraries, any further procedures contained within that nested procedure should not be called from the source library that is being reallocated or reorganized.

Parameter Summary

TO-code

Location of simulation area that contains or will contain the library. Possible codes are R1, F1, R2, and F2.

SOURCE-number (no source library in simulation area)

Create a source library. Number indicates the number of tracks you want to assign.

SOURCE-number (source library already in simulation area)

Delete or change the size of the source library. Use depends on number:

Number	Use
0	Delete
Any number but zero	Change size

SOURCE-R

Reorganize the source library.

OBJECT-number (no object library in simulation area)

Create an object library. Number indicates the number of tracks you want to assign.

OBJECT-number (object library already in simulation area)

Delete or change the size of the object library. Use depends on number:

Number	Use
0	Delete
Any number but zero	Change size

OBJECT-R

Reorganize the object library.

DIRSIZE-number

Number of tracks you want for the directory when creating, reallocating, or reorganizing the object library.

SYSTEM-NO

Do not create a scheduler work area. This will be a program simulation area.

SYSTEM-YES

Create a scheduler work area. This will be a system simulation area.

WORK-code

Location of simulation area containing space the program can use as a work area. Possible codes are R1, F1, R2, or F2.

Parameter Descriptions

TO Parameter

The TO parameter (TO-code) indicates the location of the simulation area that contains, or will contain, the library. If the program use involves both libraries, the libraries must be on the same simulation area. The TO parameter cannot be the same unit from which the library maintenance program or system is loaded. Possible codes are R1, F1, R2, and F2.

SOURCE and OBJECT Parameters

These parameters identify library uses:

Parameter	Use
SOURCE-number OBJECT-number (number is not zero)	• Create a library (if the simulation area contains no library). Number is the number of tracks you want to assign to the library. • Change the library size (if the simulation area contains a library). Number is the number of tracks you want to assign to the library.
SOURCE-0 OBJECT-0	Delete the library.
SOURCE-R OBJECT-R	Reorganize the library.

DIRSIZE Parameter

The DIRSIZE parameter allows the user to specify the size of the object library directory. The number of tracks specified (1-9) overrides the SYSTEM parameter in determining directory size. Each track can contain 288 directory entries. One entry is needed for the directory, so the formula for the number of entries in a directory is (t x 288)-1, where t is the number of tracks. If the DIRSIZE parameter is omitted, the SYSTEM parameter determines the directory size.

SYSTEM Parameter

The SYSTEM parameter applies when you create, change the size of, and reorganize object libraries. It tells the program whether you intend to include system programs in the library and create a system simulation area that can be used to perform initial program load. If system programs are to be included, a scheduler work area must be assigned.

See *Library-to-Library* under *Using the Copy Function* for information about creating a system simulation area.

Space for the scheduler work area is assigned immediately preceding the object library. If the simulation area contains a source library, the scheduler work area is between the source and object libraries. For information about the size of the scheduler work area, see index entry: scheduler work area size.

The following charts show the results of coding the SYSTEM parameter for different allocate uses.

Creating an Object Library:

Parameter	Scheduler Work Area	Directory Size[1]
SYSTEM-YES	Created	Three tracks
SYSTEM-NO	Not created	One track
Not coded	Not created	One track

[1] The directory size is overridden if the DIRSIZE parameter is coded.

Changing the Size of or Reorganizing an Object Library on a Simulation Area that Contains a Scheduler Work Area:

Parameter	Scheduler Work Area	Directory Size[1]
SYSTEM-YES	Retained	Not changed
SYSTEM-NO	Removed	Not changed
Not coded	Retained	Not changed

Changing the Size of or Reorganizing an Object Library on a Simulation Area that Does Not Contain a Scheduler Work Area:

Parameter	Scheduler Work Area	Directory Size[1]
SYSTEM-YES	Created	Not changed
SYSTEM-NO	Not created	Not changed
Not coded	Not created	Not changed

WORK Parameter

The WORK parameter (WORK-code) indicates the location of the simulation area that contains a work area. Library entries are temporarily stored in the work area while the program moves and reorganizes libraries. Possible codes are R1, F1, R2, and F2.

When the WORK parameter is coded on an ALLOCATE statement, an additional allocation of disk space may result (see index entry: allocation of disk space).

Size of the Work Area: The work area must be large enough to hold the directory and the permanent entries of the source library, object library, or both libraries depending on the program use. If you are combining uses, such as changing the sizes of both libraries, the work area must be large enough to hold the contents of both libraries.

Use	Contents of Work Area
Create a source library (simulation area contains an object library)	Object library
Change source library size (simulation area contains an object library)	Source library and object library
Change source library size (simulation area does not contain an object library	Source library
Reorganize source library	Source library
Change object library size	Object library, if not compress in place (see *Compress in Place* under *Using the Allocate Function*)
Reorganize object library	Object library, if not compress in place (see *Compress in Place* under *Using the Allocate Function*)

Location of Work Area on Disk: The program uses the first available disk area large enough to hold the library, or libraries.

Location of Simulation Area Containing the Work Area: The work area can be on either simulation area on either drive. However, it cannot be the same simulation area as the one you specified in the TO parameter. The only requirement is that the simulation area have an available area large enough for the work area. The program works faster if the simulation area containing the libraries is not on the same drive as the one containing the work area.

[1] The directory size is overridden if the DIRSIZE parameter is coded.

Using the Allocate Function

Creating a Source Library (SOURCE-number)

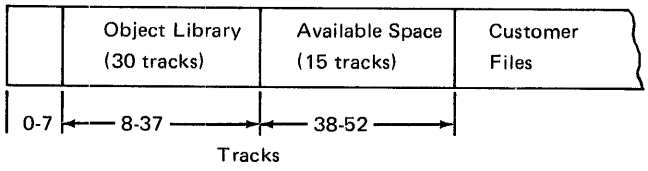
Source Library Size:

- Minimum: One track.
- Maximum: Number of tracks in the available area.
- Regardless of the number of tracks you specify, the first two sectors of the first track are assigned to the library directory. Additional sectors are used as needed for the directory.

Placement of Source Library (Simulation Area with an Object Library):

- The source library must immediately preceed the object library. A disk area large enough for the source library must follow the object library because the program moves the object library to make room for the source library (Figure 49). To do this, it needs a work area. (See *WORK Parameter.*) The object library is reorganized, and all temporary entries are deleted.
- If you allocate a source library after deleting it, the program automatically moves the object library to make room for the source library. The starting location of the source library is the previous starting location of the object library.

Disk Space Before Creating Source Library



Disk Space After Creating Source Library

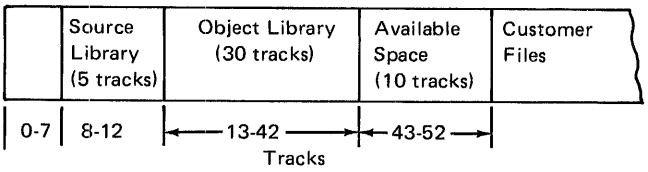


Figure 49. Moving Object Library to Insert Source Library

Placement of the Source Library (Simulation Area without an Object Library): The program assigns the source library to the first available disk area large enough for the library.

If you allocate a source library after deleting it, the source library is assigned the same way.

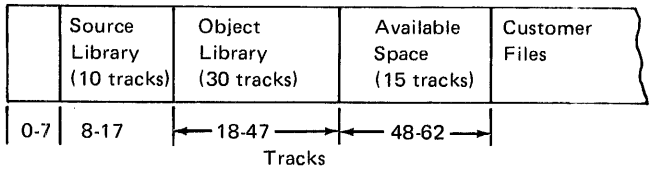
Changing the Size of (Reallocating) a Source Library (SOURCE-number)

Any time the program changes the source library size, it reorganizes both the source and object libraries and deletes all temporary entries. (See *Reorganizing a Source Library* under *Using the Allocate Function.*) To do this, it needs a work area. (See *WORK Parameter.*)

Making the Source Library Larger:

- If the simulation area contains an object library, space must be available immediately following the object library. The program moves the object library to make tracks available at the end of the source library (Figure 50).
- If the simulation area does not contain an object library, space must be available immediately following the source library.

Disk Space Before Tracks Are Added to Source Library



Disk Space After 5 Tracks Are Added to Source Library

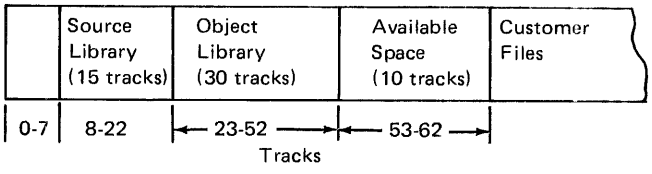
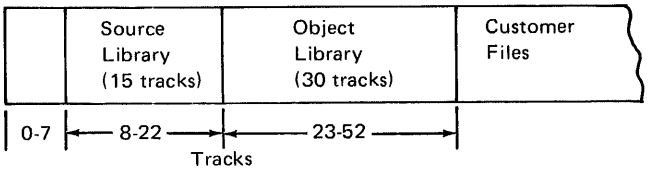


Figure 50. Increasing Source Library Size

Making the Source Library Smaller:

- If the simulation area contains an object library, the program moves the end location of the source library to make the library smaller. The object library is moved and space becomes available following the object library (Figure 51).
- If the simulation area does not contain an object library, the program moves the end location of the source library to make the source library smaller.

Disk Space Before Source-Library Size was Decreased



Disk Space After 5 Tracks Were Taken From Source Library

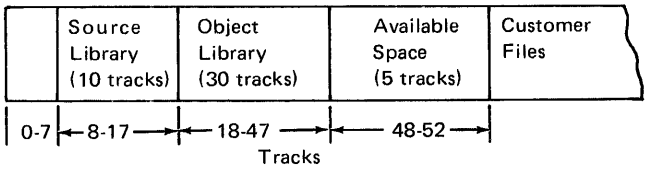
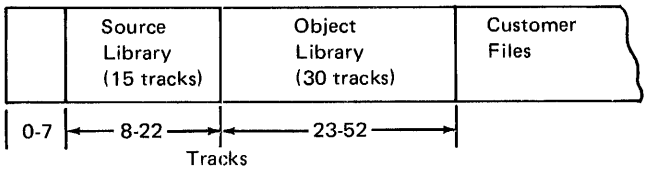


Figure 51. Decreasing Source Library Size

Deleting a Source Library (SOURCE-O)

The program makes the disk area occupied by the source library available for other use (disk files). (See Figure 52.)

Disk Space Before Source Library Deleted



Disk Space After Source Library Deleted

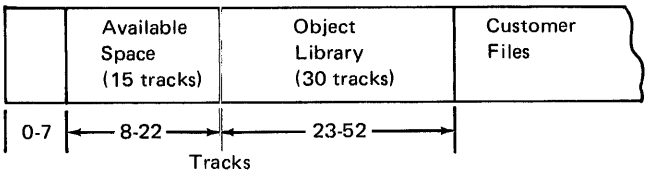


Figure 52. Deleting Source Library

Reorganizing a Source Library (SOURCE-R)

Reason for Reorganizing the Library: Areas from which source library entries are deleted are completely reused for new entries. If an entry exceeds the space in such an area, the program puts as much of the entry as will fit in the area and continues the entry in the next available area. In this way, the program efficiently uses library space. This can, however, decrease the speed at which those entries can be read from the library. Therefore, if you frequently add and delete source library entries, you should reorganize your source library periodically.

Reorganizing the Library: The program relocates entries so that no entry is started in one area and continued in another. All temporary entries are deleted. The program needs a work area. (See *WORK Parameter.*)

Creating an Object Library (OBJECT-number)

Object Library Size:

- Minimum: Three tracks including the directory tracks.
- Maximum: Number of tracks in available area.
- Library directory: The first 3 tracks in the library are reserved for the library directory if the library is to contain system programs; otherwise, only the first track is used. If the DIRSIZE parameter is entered, the directory size specifed is used.
- Scheduler Work Area: The scheduler is a component of the System/3 SCP that reads and processes OCL statements. It uses a work area on the simulation area called the scheduler work area (SWA), to temporarily save OCL file label information during the processing of a program. The area is allocated when SYSTEM-YES is specified. The work space is not included in the number you specify in the OBJECT parameter; the space is calculated and assigned by the library maintenance program. The amount of space needed depends on whether DPF (dual program feature), checkpoint/restart and/or the inquiry capability is supported. For non-DPF systems, 2 tracks are needed; for DPF systems, 4 tracks are needed. The inquiry and checkpoint/restart features require additional tracks for a rollout/rollin area. The number of tracks needed depends on the main storage size of the system.

Main Storage Size	Rollout/Rollin Tracks
32K	7
48K	9
64K	12

The SWA contains simulation area usage information, F1 and F7 label information, an initiator table, utility control statement area, and miscellaneous work areas. There is one SWA for each program level. (See *Maximum Number of Files, IBM System/3 Model 12 User's Guide,* GC21-5142.)

Placement of Object Library (Simulation Area With a Source Library): Space for the object library must be available immediately following the source library.

Placement of Object Library (Simulation Area Without a Source Library): The program assigns the object library to the first available disk area that is large enough.

Changing the Size of (Reallocating) an Object Library (OBJECT-number)

Making the Library Larger: The number of tracks you want to add must be available immediately following the object library. The program assigns the additional tracks to the library. (The starting location of the library remains unchanged.)

Making the Library Smaller: The program moves the end location of the object library to decrease the library size. Tracks, therefore, become available following the library.

Reorganizing the Library: Any time the program changes the library size, it also reorganizes the library and deletes all temporary entries. (See *Reorganizing an Object Library.*) A work area is needed if other functions are being performed with the reorganization. (See *WORK Parameter.*) If not, a work area is not used. (See *Compress in Place* under *Using the Allocate Function.*)

Deleting an Object Library (OBJECT-0)

The program makes the disk area occupied by the object library (and the scheduler work area if this was a system simulation area) available for other use.

Reorganizing an Object Library (OBJECT-R)

Gaps can occur between object library entries when you add and delete entries. By reorganizing the library, these gaps are removed. When the library is reorganized, all temporary entries are deleted. A work area is needed if other functions are being performed with the reorganization. (See *WORK Parameter.*) If not, a work area is not used. (See *Compress in Place* under *Using the Allocate Function.*)

Compress in Place $\left[\text{OBJECT} - \left\{\begin{matrix} R \\ number \end{matrix}\right\}\right]$

If an object library is to be reorganized, or the size is to be changed and this is the only function to be performed, the object library is compressed in place. This means that the library is reorganized with all gaps removed and all temporary entries deleted without the use of a work area. The WORK parameter is ignored if supplied.

If, however, a source library function is to be performed or if the directory size (DIRSIZE parameter) or the simulation area type (SYSTEM parameter) is to be changed in conjunction with an object library function, a work area will be used. (See *WORK Parameter.*)

COPY FUNCTION

Uses

Reader-to-Library	Add or replace a library entry. The reader is the system input device.
File-to-Library	Add or replace one or more library entries. A disk file is the input.
Library-to-File	Copy one or more library entries to a disk file.
Library-to-Library	Copy one library entry (or those entries with the same name from all libraries).
	Copy library entries that have names beginning with certain characters.
	Copy all library entries.
	Copy minimum system.
Library-to-Printer	Print one library entry (or those entries with the same name from all libraries).
	Print library entries that have names beginning with certain characters.
	Print all library entries of a certain type.
	Print directory entries for library entries of a certain type.
	Print entries from all directories including system directory.
	Print system directory only.
Library-to-Card	Punch one library entry (or those entries with the same name from all libraries).
	Punch library entries that have names beginning with certain characters.
	Punch all library entries of a certain type.
Library-to-Printer and-Card	Print and punch one library entry (or those entires with the same name from all libraries).
	Print and punch library entries that have names beginning with certain characters.
	Print and punch all temporary or permanent library entries of a certain type.

Control Statement Summary

Reader-To-Library

Add or Replace a Library Entry:

// COPY FROM-READER,LIBRARY-{S|P|O|R},NAME-name,TO-code,RETAIN-{T|P|R}

Library Entry

// CEND Must always follow the source or object entry being placed into the source or object library.

/* or /& statements cannot be present in the entries being copied into the libraries.

File-To-Library

Add or Replace One or More Library Entries:

// COPY FROM-DISK,FILE-filename,RECL-{80|96},TO-code,RETAIN-{R|P}

Example of Data in Disk File:

```
// COPY FROM-READER,LIBRARY-O,RETAIN-P,NAME-DECK01①
  -
  -
  load module
  -
  -
// CEND
// COPY LIBRARY-S,NAME-DECK02①
  -
  -
  source module
  -
  -
// CEND
// END②
```

① Only the LIBRARY and NAME parameters are required. Other parameters are ignored.

② The // END statement read from the file is optional. It causes the next statement to be read from the system input device or procedure. A // END statement must still be read from the system input device or procedure to indicate the end of the library maintenance control statements.

Library-To-File

Copy One or More Library Entries to a File:

// COPY FROM-code,TO-DISK,FILE-filename,RECL-{80 / 96}

Control Statements Following // COPY:

// ENTRY LIBRARY-{S / P / O / R / ALL},NAME-{name / characters.ALL / ALL} ①

// NEND (Required to terminate the copy to file.)

---

① Any number of // ENTRY statements may precede the // NEND statement.

Library-To-Library

Copy One Library Entry (or Entries with the Same Name from All Libraries):

// COPY FROM-code,LIBRARY-{S/P/O/R/ALL},NAME-name,TO-code,RETAIN-{T/P/R},NEWNAME-name①

Copy Library Entries that Have Names Beginning with Certain Characters:

// COPY FROM-code,LIBRARY-{S/P/O/R/ALL},NAME-characters.ALL,TO-code,RETAIN-{T/P/R},NEWNAME-characters①

Copy All Library Entries:

// COPY FROM-code,LIBRARY-{S/P/O/R/ALL},NAME-ALL,TO-code,RETAIN-{T/P/R}②

Copy Minimum System:

// COPY FROM-code,LIBRARY-O,NAME-SYSTEM,TO-code②

---

①NEWNAME parameter is needed in any of the following cases:

- If you want the copy to have a different name than the original entry.
- If you want to replace an entry on the TO unit with an entry from the FROM unit, but the entries have different names.
- If you want the names of the copies to begin with different characters than the names of the original entries. The same number of characters must be in the NEWNAME parameter as in the NAME parameter.
- If the FROM and TO units are the same.

Note: NEWNAME cannot be DIR,ALL, or SYSTEM.

②FROM and TO parameters cannot be the same unit.

Library-To-Printer-and/or-Card

Print and/or Punch One Library Entry (or Entries with the Same Name from All Libraries):

```
// COPY FROM-code,LIBRARY-{S|P|O|R|ALL},NAME-name,TO-{PUNCH|PRINT|PRTPCH}
```

Print and/or Punch Temporary and Permanent Library Entries that Have Names Beginning with Certain Characters:

```
// COPY FROM-code,LIBRARY-{S|P|O|R|ALL},NAME-characters.ALL,TO-{PUNCH|PRINT|PRTPCH}
```

Print and/or Punch All Temporary and Permanent Library Entries of a Certain Type:

```
// COPY FROM-code,LIBRARY-{S|P|O|R},NAME-ALL,TO-{PUNCH|PRINT|PRTPCH}
```

Print Directory Entries for Library Entries of a Certain Type:

```
// COPY FROM-code,LIBRARY-{S|P|O|R},NAME-DIR,TO-PRINT
```

Print Entries from All Directories Including System Directory:

```
// COPY FROM-code,LIBRARY-ALL,NAME-DIR,TO-PRINT
```

Print System Directory Entries Only:

```
// COPY FROM-code,LIBRARY-SYSTEM,NAME-DIR,TO-PRINT
```

Print Directory Entries, Omitting Selected Entries:

```
// COPY FROM-code,LIBRARY-{S|P|O|R|ALL},NAME-DIR,TO-PRINT,OMIT-{name|characters.ALL}
```

Parameter Summary

FROM-READER

Entry to be placed in library is to be read from system input device.

FROM-code

Location of simulation area containing library entries being copied, printed, or punched. Possible codes are R1, F1, R2, and F2.

FROM-DISK

The entry or entries to be placed into a library or libraries reside in a disk file. The disk file must be described by an OCL FILE statement.

FILE-filename

For a file-to-library or library-to-file copy, this parameter is needed to identify the file on disk. The filename must match the filename on the OCL FILE statement.

RECL-{80 / 96}

For a file-to-library or library-to-file copy, this parameter gives the size of the disk records. Only 80- or 96-column card image records are allowed. If this parameter is omitted, 96 is assumed.

LIBRARY-{S / P / O / R}

Type of library entries involved in copy use. Possible codes are:

Code	Meaning
S	Source statements (source library)
P	OCL procedures (source library)
O	Object programs (object library)
R	Routines (object library)

LIBRARY-ALL

All types of entries (S, P, O, and R) from both libraries are involved in copy use.

LIBRARY-SYSTEM

Only system directory entries are being printed.

NAME-{name / characters.ALL / ALL}

Specific library entries on the FROM unit, of the type indicated in LIBRARY parameter, involved in copy use. Possible information is:

Information	Meaning
name	Name of the library entry involved.
characters.ALL	Only those entries beginning with the indicated characters. For example, $MA.ALL means the library maintenance program ($MAINT).
ALL	All entries. (The type indicated in LIBRARY parameter).

NAME-SYSTEM

System programs that make up the minimum system and IPL information contained on cylinder 0 are copied. The minimum system is made up of system programs necessary to load and run programs. System programs necessary to generate and maintain the system such as utilities are not included in the minimum system.

NAME-DIR

Directory entries for all library entries of the type indicated in the LIBRARY parameter are involved in the copy use. If the LIBRARY parameter is LIBRARY-ALL, system directory entries are also printed.

RETAIN-{T / P / R}

Adding Entry to Library. RETAIN gives designation of the TO entry:

Code	Meaning
T	Temporary
P or R	Permanent

Replacing Existing Library Entry. RETAIN gives designation of the TO entry and tells program whether to halt before replacing entry:

Code	Meaning
T	Temporary designation. Halt before replacing entry.
P	Permanent designation. Halt before replacing entry.
R	Permanent designation. Do not halt before replacing entry.

Printing or Punching Entries. The RETAIN parameter is ignored.

TO-code

Location of simulation area that is to contain the copies of the entries. Possible codes are R1, F1, R2, and F2.

TO-PRINT

Entries are printed.

TO-PUNCH

Entries are punched.

TO-PRTPCH

Entries are printed and punched.

TO-DISK

The entry or entries are to be copied to a disk file. The disk file must be described by an OCL FILE statement.

NEWNAME-name

Name you want used on the TO unit to identify the entries put on that simulation area. If you omit this parameter, the program uses the NAME parameter in naming the entries.

NEWNAME-characters

Beginning characters you want to use in names identifying entries being put on the TO unit. You must use the same number of characters as in the NAME parameter (NAME-characters.ALL). If you omit this parameter, the program uses the NAME parameter in naming the entries.

OMIT-name

When printing directory entries, omit the entry specified by *name*.

OMIT-characters.ALL

When printing directory entries, omit all entries with these beginning characters.

Library Directories

Source and Object Library Directories

- The source and object libraries have separate library directories. Every library entry has a corresponding entry in its library directory. The directory entry contains such information as the name and location of the library entry (see Figures 53 through 55).

- The library maintenance program makes entries in the directories when it puts entries in the libraries.

System Directory

- Every simulation area that contains libraries contains a system directory. The system directory contains information about the sizes of and available space in libraries and their directories (see Figures 53 through 55).

- The library maintenance program creates and maintains the system directory.

Naming Library Entries

Characters to Use

Use any combination of System/3 characters except blanks, commas, quotes, and periods. (Appendix A lists the characters.) The names of most IBM programs begin with a dollar sign ($). Therefore, to avoid possible duplication, do not use a dollar sign as the first character in the names you use for your entries. The first character must be alphabetic.

Length of Name

The name can be from 1 to 6 characters long.

Restricted Names

Do not use the names ALL,DIR, and SYSTEM. They have special meanings in the NAME parameter.

Entries with the Same Name

For each of the two physical libraries, source and object, there are two types of entries. The source library has type P and type S entries. The object library has type O and type R entries. Entries of the same type cannot have the same name, but entries of different types may. For example, two procedures in a source library cannot have the same name, but a procedure and a set of source statements can.

Retain Types

Temporary Entries

- Temporary entries are entries you do not intend to keep in your libraries. They are normally used only once or a few times over a short period.

- In the object library, temporary entries are placed together following the permanent entries. Any time a permanent entry is added to the library, all temporary entries are deleted. Temporary entries are also deleted when you replace one permanent entry with another.

- In the source library, temporary and permanent entries can be in any order. One entry is placed after another regardless of their designations. Temporary entries, therefore, are not automatically deleted every time you add a permanent entry. However, when the source library is reallocated or reorganized, only permanent entries remain.

- You can use temporary entries as often as you like until they are deleted.

- A temporary entry cannot replace a permanent entry.

Permanent Entries

- Permanent entries are entries you intend to keep in your libraries. They are normally entries you use often or at regular intervals (once a week, once a month, and so on).

- The program does not delete permanent entries unless you use the delete function of library maintenance to delete them, or the allocate function to delete the entire library.

Using the Copy Function

Reader-to-Library

Input: The program reads one library entry. It can be any one of the following types:

Source statements

Procedure

Object program

Routine

The entry is read from the system input device.

The header card on an object deck (H in column 1) contains the date the deck was punched. This date is in columns 58-63 and is in the format of the system date, either mmddyy or ddmmyy.

Output:

- Duplicate characters are removed from source statements and procedures before they are put in the source library. The program does not check them for errors.
- Object programs and routines are placed in the object library after sequence and checksum information is removed.

Adding Entires: The program can add a new entry to a library. The name of the entry is taken from the NAME parameter. See *Naming Library Entries* for valid names. The RETAIN parameter specifies whether the entry will be temporary or permanent. If the RETAIN parameter is omitted, RETAIN-T is assumed (See *Retain Types.*)

Replacing Existing Entries:

- The program can replace an existing library entry with the entry you are putting in the library. The RETAIN paramater specifies the new retain type. If the RETAIN parameter is omitted, RETAIN-T is assumed. A temporary entry cannot replace a permanent entry.
- The program can halt before replacing an existing entry. Whether it does depends on the RETAIN parameter you use (See *RETAIN Parameter.*)
- Before the new entry is added, the duplicate entry is deleted. Additional library space is not needed unless the new entry is larger than the old one.

File-to-Library

Input: The disk file can contain one or more library entries. The entries must be in the format put out by the library-to-card function or by the linkage editor. The // COPY statement at the beginning of each entry contains the name of the entry and the type of library (S, P, O, R). A // CEND statement must follow each entry in the file.

The disk file must be a sequential file and be defined by a FILE statement in the OCL for the library maintenance program. Multivolume files are not supported.

Output: The output from the file-to-library function is the same as for the reader-to-library function except that temporary entries are not allowed.

Library-to-File

Input: The program can copy one or more library entries from a library to a disk file. The types of entries can be:

Source statements

Procedures

Object programs

Routines

All of the preceding types

The NAME and LIBRARY parameters on the // ENTRY statements specify which entries to copy. A single library-to-file function must be the only valid function performed within a LOAD-RUN of the library maintenance program.

Output: The output from the library-to-file function has the same format as for the library-to-card function. The output is written to a sequential disk file defined by an OCL FILE statement and created by the library maintenance program. Multivolume files are not supported.

Library-to-Library

Input: The program can copy one or more library entries from one simulation area to another. The types of entries can be:

Source statements

Procedures

Object programs

Routines

All of the preceding types

Minimum system

The NAME and LIBRARY parameters specify which entries to copy.

Output:

- The entries, regardless of their type, are copied from one simulation area to the other without change.
- The NEWNAME parameter is used to copy and rename entries on the same simulation area. (See *NEWNAME Parameter* and *Naming Library Entries.*)
- The RETAIN parameter specifies whether the entries are to be temporary or permanent. If the RETAIN parameter is omitted, RETAIN-T is assumed. When the parameters LIBRARY-ALL and NAME-ALL or LIBRARY-O and NAME-SYSTEM are used, RETAIN-P is assumed and RETAIN-T is invalid.
- Copying a minimum system (LIBRARY-O, NAME-SYSTEM) or copying all of the types (LIBRARY-ALL, NAME-ALL) are the functions used to create a system simulation area that can be used to perform initial program load. (Copying LIBRARY-ALL, NAME-ALL creates a system simulation area only if the FROM area is a system simulation area.) Because of this use, the object library on the simulation area you specify in the TO parameter must be empty. (It cannot contain any entries or deleted entries.) Also the object library on the TO area must have been allocated with a scheduler work area and a rollout/rollin area at least as large as those on the FROM simulation area.

Adding Entries:

- You can omit the NEWNAME parameter. If you do, the name used for the copy is taken from the NAME parameter. (The copy has the same name as the original entry.)
- If NAME-ALL is specified, the names by which the entries are identified on the FROM simulation area are also used on the TO simulation area to identify the entries.

Replacing Existing Entries:

- The program can replace existing entries with the entries you are putting in the library. If the entry you are copying (the entry in the simulation area you identify in the FROM parameter) has the same name as the entry you are replacing (the entry in the simulation area you identify in the TO parameter), you must omit the NEWNAME parameter because the NEWNAME parameter cannot be the same as the NAME parameter. If the names are not the same, you must use the NEWNAME parameter to give the name of the entry being replaced.

- The program can halt before replacing an existing entry. Whether it does depends on the RETAIN parameter. (See *RETAIN Parameter.*)

- A temporary entry cannot replace a permanent entry.

Library-to-Print and/or Card

Types of Entries that Can be Printed or Punched:

- The program can print or punch one or more library entries. They can be any one of the following types:

  Source statments

  Procedures

  Object programs

  Routines

  All of the preceding types (limited to entries having the same name or entries beginning with the same characters)

- The program can print (but not punch) the following types of directory entries:

  Source statements

  Procedures

  Object programs

  Routines

  System directory

  All of the preceding types

The program sorts directory names before printing them only if there is available work space on the FROM simulation area. This causes an allocation of disk space. (See *Allocation of Disk Space* under *Using the Allocate Function.*)

Printed or Punched Library Entires:

- Duplicate characters are reinserted into source statements and procedures to make them readable.

- Object programs and routines are printed and punched after sequence information and checksum information (punch only) has been added.

- The library entries when punched, are preceded by a // COPY statement of the reader-to-library format and followed by a // CEND statement.

Printout of Directory Entries

- The format of the source library directory printout is described in Figure 53. If there is no source library in the simulation area, the message NO SOURCE LIBRARY EXISTS is printed. If a source library exists but is empty, the NO SOURCE DIR ENTRIES EXIST message is printed.

- The format of the object library directory printout is described in Figure 54. If there is no object library in the simulation area, the message NO OBJECT LIBRARY EXISTS is printed. If an object library exists but is empty, the NO OBJECT DIR ENTRIES EXIST message is printed.

- A sample system directory printout is described in Figure 55. If there is no source library in the simulation area, the message NO SOURCE LIBRARY EXISTS ON THIS PACK is printed. If there is no object library in the simulation area, the message NO OBJECT LIBRARY EXISTS ON THIS PACK is printed.

```
SOURCE DIRECTORY FROM XX VOL. ID XXXXXX MM/DD/YY

                      ADDRESS
TYPE   NAME   FIRST@   LAST@   ATTRI   #SECTORS
 X    XXXXXX  XXX-XX   XXX-XX    X       XXXX
```

Explanation:

Heading	Meaning
TYPE	S = source statements P = procedure
NAME	Name of library entry (up to 6 characters)
ADDRESS (FIRST and LAST)	Addresses of first and last sectors that contain the library entry. Addresses are expressed by track and sector numbers. Example: 008-03 means track 8, sector 3.
ATTRI	T = temporary P = permanent
#SECTORS	Total number of sectors for the library entry.

Figure 53. Source Library Directory Printout

OBJECT DIRECTORY FROM XX VOL. ID XXXXX MM/DD/YY

TYPE		NAME	DSK ADD	CYL/ SEC	TXT-CAT	LINK ADDR	RLD DISP	ENTRY PNT	CORE SEC	ATTR	LEVEL	TOT SEC
X	X	XXXXXX	TTT/SS	CC/SS	XXX	XXXX	XX	XXXX	XXX	XXXX	XXX	XXXX

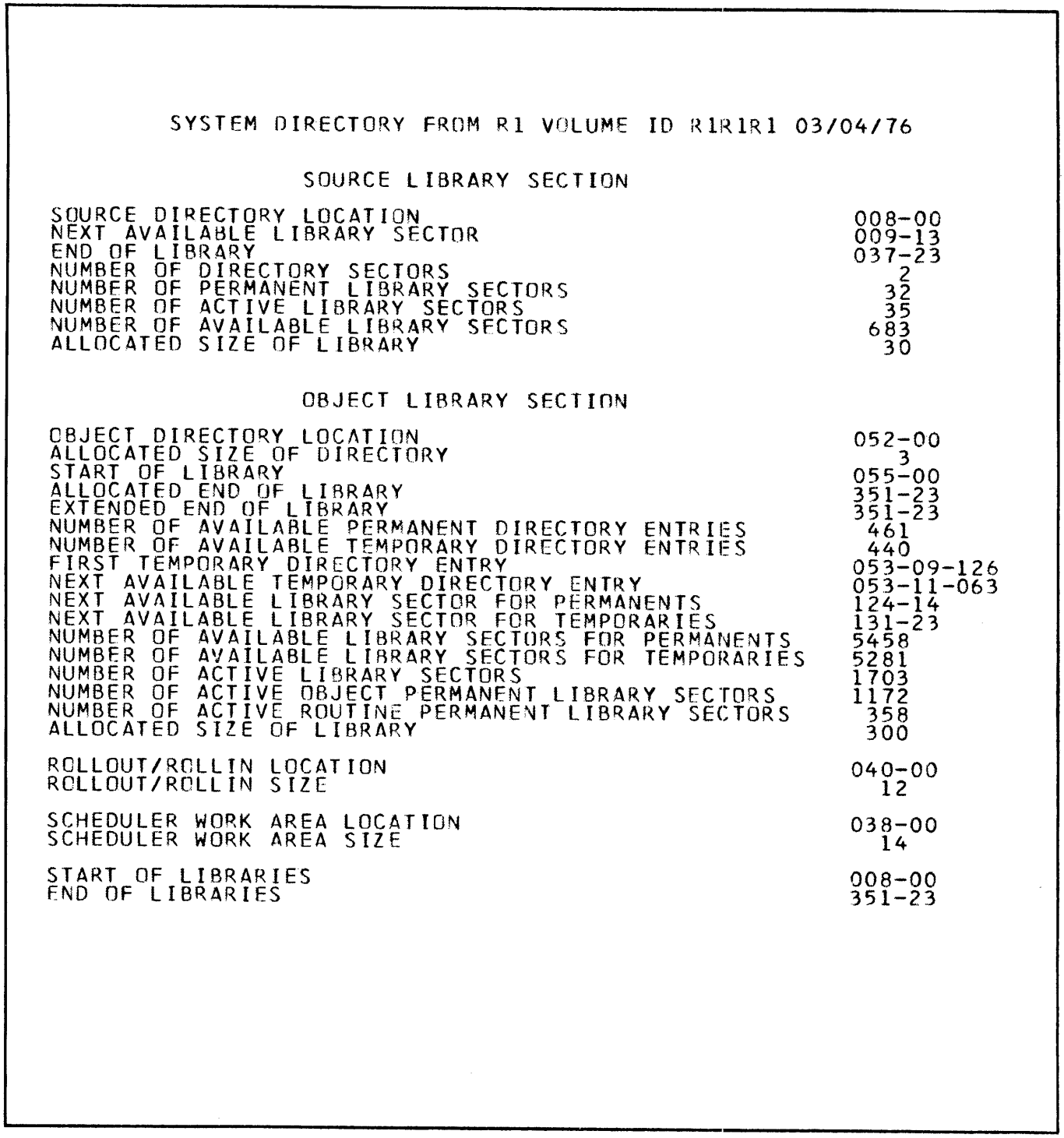
Explanation:

Heading	Meaning
TYPE	The first character printed indicates the attributes of the entry as follows: P = permanent T = temporary The second character printed indicates the type of module the entry is. Its meaning is as follows: O = Object program R = routine
NAME	Name of library entry (up to 6 characters)
DSK ADD	Address where library entry begins on disk. Example: 015/10 means track 15, sector 10 (in decimal). T = track, S = sector.
CYL/SEC	Address where library entry begins on disk (in hexadecimal). C = cylinder, S = sector.
TXT-CAT	For object programs, this number indicates the number of sectors used for the text portion of the library entry. Object programs consist of two parts: text and RLD. Text is the program; RLD is information used in loading the program for execution. For routines, this number is the category of the routine. This number is used by the overlay linkage editor for determining overlays.
LINK ADDR	Object programs only. Assigned core hexadecimal address of this library entry.
RLD DISP	Object programs only. It indicates the hexadecimal position in which RLD information begins in the last text sector. If the last text sector contains no RLD information, the RLD displacement is 0, indicating the information starts in the next sector.
ENTRY PNT	Object programs only. Main storage address (hexadecimal) where program execution begins before relocations.
CORE SEC	Core size, given in sectors, required to run the program.

Figure 54. (Part 1 of 2). Object Library Directory Printout

Heading	Meaning
ATTR	Byte 1: Bit 0=1 Permanent entry. 0 Temporary entry. Bit 1=1 Inquiry. This program requires that the Request key be pressed to start processing. Bit 2=1 Inquiry invoking. This program runs in program level 1 and can be rolled out to allow an Inquiry program to run. Bit 3=1 Dedicated. In a DPF system, this program must run with the other program level inactive. Bit 4=1 Source required. This program requires the allocation of the $WORK and $SOURCE files. $SOURCE must be filled either from the system input device or a source library. Bit 5=1 Deferred mount. This program accepts mounting of data modules during its execution. Bit 6=1 PTF applied. A program temporary fix (PTF) has been applied to this program. Bit 7=1 Overlay object program. Byte 2: Bit 0=1 System input dedication. The system input device must be dedicated to this program. The device may be released when no longer needed. Bit 1=1 Checkpoint/restart program. Bit 2=1 Direct source read. This program can have a // COMPILE statement and a no source required attribute (byte 1, bit 4=0). The program accesses the source library itself. Bit 3=1 Macro processor allowed. This program can be preceded by the macro processor. If the source required attribute is present and a // SWITCH 1XXXXXXX statement was processed, the $SOURCE file is opened as input instead of output. Bit 4 Reserved. Bit 5=1 Program common. This program requires that a new load address be calculated at load time to place it in main storage beyond its own program common region. Bit 6=1 Model 12 compile. Bit 7 Reserved.
LEVEL	Release level of system programs. For user programs this can be assigned by the overlay linkage editor.
TOT SEC	Total number of disk sectors occupied by the library entry.

Figure 54. (Part 2 of 2). Object Library Directory Printout



```
SYSTEM DIRECTORY FROM R1 VOLUME ID R1R1R1 03/04/76

SOURCE LIBRARY SECTION

SOURCE DIRECTORY LOCATION                                  008-00
NEXT AVAILABLE LIBRARY SECTOR                              009-13
END OF LIBRARY                                             037-23
NUMBER OF DIRECTORY SECTORS                                     2
NUMBER OF PERMANENT LIBRARY SECTORS                            32
NUMBER OF ACTIVE LIBRARY SECTORS                               35
NUMBER OF AVAILABLE LIBRARY SECTORS                           683
ALLOCATED SIZE OF LIBRARY                                      30

OBJECT LIBRARY SECTION

OBJECT DIRECTORY LOCATION                                  052-00
ALLOCATED SIZE OF DIRECTORY                                     3
START OF LIBRARY                                           055-00
ALLOCATED END OF LIBRARY                                   351-23
EXTENDED END OF LIBRARY                                    351-23
NUMBER OF AVAILABLE PERMANENT DIRECTORY ENTRIES               461
NUMBER OF AVAILABLE TEMPORARY DIRECTORY ENTRIES               440
FIRST TEMPORARY DIRECTORY ENTRY                            053-09-126
NEXT AVAILABLE TEMPORARY DIRECTORY ENTRY                   053-11-063
NEXT AVAILABLE LIBRARY SECTOR FOR PERMANENTS               124-14
NEXT AVAILABLE LIBRARY SECTOR FOR TEMPORARIES              131-23
NUMBER OF AVAILABLE LIBRARY SECTORS FOR PERMANENTS            5458
NUMBER OF AVAILABLE LIBRARY SECTORS FOR TEMPORARIES           5281
NUMBER OF ACTIVE LIBRARY SECTORS                              1703
NUMBER OF ACTIVE OBJECT PERMANENT LIBRARY SECTORS             1172
NUMBER OF ACTIVE ROUTINE PERMANENT LIBRARY SECTORS             358
ALLOCATED SIZE OF LIBRARY                                      300

ROLLOUT/ROLLIN LOCATION                                    040-00
ROLLOUT/ROLLIN SIZE                                            12

SCHEDULER WORK AREA LOCATION                               038-00
SCHEDULER WORK AREA SIZE                                       14

START OF LIBRARIES                                         008-00
END OF LIBRARIES                                           351-23
```

Figure 55. (Part 1 of 2). System Directory Printout

Using the System Directory to Determine if the Object Library Should Be Reorganized

The following are *not* updated when an object library entry is deleted:
- Number of available directory entries.
- Next available directory entry.
- Next available library sector.
- Number of available library sectors.

These reflect only contiguous space that can be used, therefore, gaps are not included. (See *Object Library* under *Organization of Library Entries.*)

To calculate the total number of sectors that could be made available for permanent entries if the object library is reorganized, perform the following procedure. Take values from Figure 55 (Part 1 of 2).

1. Determine the object library size in sectors	Allocated size of library	=	300
	Allocated size of directory	=	− 3
	Object library size (tracks)	=	297
			x24
	Object library size (sectors)	=	7128
2. Determine the number of permanent object library sectors	Number of active object permanent library sectors	=	1172
	Number of active routine permanent library sectors	=	+358
	Number of permanent object library sectors	=	1530
3. Determine the number of contiguous sectors that will be available at the end of the library if the library is reorganized to remove all gaps and temporary library entries	Object library size (sectors) from step 1	=	7128
	Number of permanent object library sectors from step 2	=	−1530
	Number of available sectors	=	5598
4. Compare the number of available sectors calculated to the number of available library sectors for permanents	Number of available sectors from step 3	=	5598
	Number of available library sectors for permanents	=	−5458
	Difference in sectors	=	140

This difference (140) represents the amount of contiguous space that can be gained by reorganizing the object library.

Figure 55. (Part 2 of 2). System Directory Printout

DELETE FUNCTION

Uses

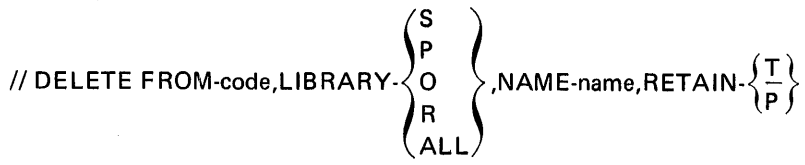
- Delete a temporary of permanent entry from a library (or entries with the same name from all libraries).
- Delete temporary or permanent library entries that have names beginning with certain characters.
- Delete all temporary or permanent library entries of a certain type.

Considerations and Restrictions

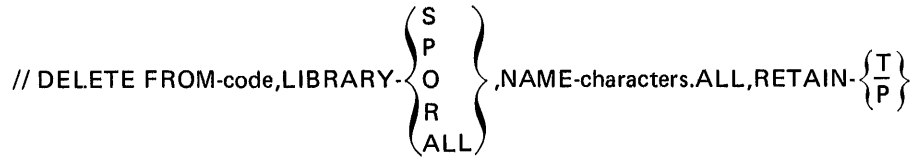
- System modules cannot be deleted from the active system simulation area (the simulation area the system was loaded from at IPL time).
- Library maintenance program modules cannot be deleted from the active program simulation area.
- When all temporary entries are deleted from the object library using LIBRARY-O,NAME-ALL,RETAIN-T, the temporary routines (LIBRARY-R) are also deleted.
- The RETAIN parameter must match the attribute of the entry in the library. Otherwise, the entry is considered not found. RETAIN-T is assumed if the RETAIN parameter is omitted.

Control Statement Summary

Delete a Temporary or Permanent Library Entry (or Entries with the Same Name from All Libraries)



Delete Temporary or Permanent Entries with Names Beginning with Certain Characters



Delete All Temporary or Permanent Entries of a Certain Type

Parameter Summary

FROM-{R1 / F1 / R2 / F2}

Location of simulation area that contains library entries you are deleting. Possible codes are R1, F1, R2, and F2.

LIBRARY-{S / P / O / R / ALL}

Type of entries being deleted. Possible codes are:

Code	Meaning
S	Source statements (source library)
P	Procedures (source library)
O	Object programs (object library)
R	Routines (object library)
ALL	All types of entries (S, P, O, and R) are being deleted

NAME-{name / characters.ALL / ALL}

Particular entries, of type indicated in LIBRARY parameter, being deleted. These entries are further identified by the RETAIN parameter. Possible codes are:

Code	Meaning
name	Name of the library entry, or entries, being deleted.
characters.ALL	Entries that have names beginning with the indicated characters. You can use up to 5 characters. Example: NAME-INV.ALL refers to the entries having names that begin with INV.
ALL	All entries (of the type indicated in LIBRARY parameter). NAME-ALL cannot be used with LIBRARY-ALL.

RETAIN-{T / P}

Designation of entries being deleted:

Code	Meaning
T	Temporary
P	Permanent

MODIFY FUNCTION

Uses

- Maintain source statements and procedures by using a card reader.
- Reserialize a source library entry.
- List the statements in a source library entry.
- Remove statements from a source library entry.
- Replace source library statements.
- Insert statements into a source library entry.

Considerations and Restrictions

- Sequence numbers are a physical part of the source record and must be placed where they cannot conflict with other data in the record. In a procedure they should be placed near the end of the record beyond the OCL and utility control statements' keywords and parameters. The sequence numbers should be placed in source statements where they do not overlay data. For example, data could be destroyed if sequence numbers were placed in RPG II source statements that contained compile-time tables.
- At least three control statements must be entered to modify the source library. A // MODIFY statement is needed to describe the library entry. A // REMOVE, // REPLACE, or // INSERT statement describes the type of modification. A // CEND statement indicates the end of the modify control statements.
- The simulation area specified by the WORK parameter on the // MODIFY statement must contain a work area large enough to hold the modified source library entry.
- The sequence numbers specified by the FROM-seqno, TO-seqno, and AFTER-seqno parameters on the // REMOVE, // REPLACE, and // INSERT statements must be valid numbers and exist in the source library entry. There are no default values for these parameters. The number of digits entered must be the same as the number of positions specified by the SEQFLD parameter.
- All statements in a source library entry must have ascending sequence numbers in the positions specified by the SEQFLD parameter.
- Multiple operations (REMOVE, REPLACE, INSERT) may be performed within the same MODIFY run if they are done in an ascending sequential order. That is, the FROM sequence number in a REMOVE or REPLACE statement must be greater than the last sequence number in the preceding statement. The AFTER sequence number of an INSERT statement must be equal to or greater than the last sequence number of the preceding statement. Consecutive INSERT statements must not have the same sequence number.
- When modification is complete, the directory entry is written back with a permanent attribute.
- The control statements following the // MODIFY statement are read from the system input device.
- Since the REMOVE control statement is valid for both the $DELET system utility and $MAINT system utility, care should be used when modifying a $DELET procedure. The program attempts to determine whether the REMOVE statement is data or a control statement. If a determination cannot be made, the program halts and waits for further instructions.
- If LIST-YES is specified and a printer error (causing a halt) occurs during the listing of the source library entry, responding to the halt with a 2-option causes the listing to stop. The modified entry is then placed back in the library before the function is terminated with a controlled cancel.

Control Statement Summary

Initiate Modification

// MODIFY NAME-name,FROM-code,LIBRARY-{S|P},WORK-code,RESER-{YES|NO|ONLY},LIST-{YES|NO},

SEQFLD-xxyy,INCR-number

Control Statements Following // MODIFY

Delete all statements between and including the FROM and TO sequence numbers.

// REMOVE FROM-seqno, TO-seqno

Replace all statements between and including the FROM and TO sequence numbers with the statements supplied:

// REPLACE FROM-seqno, TO-seqno
-
-
1 - n statements to replace those removed
-

Insert the supplied statements after the statement indicated by the AFTER parameter:

// INSERT AFTER-seqno
-
1 - n statements to be inserted
-

// CEND must follow the control statements to terminate the modify function.

Parameter Summary

NAME-name

Name of the entry you are modifying. This is the name that identifies the entry in the library directory.

FROM-code

Location of the simulation area that contains the entry you are modifying. Possible codes are R1, F1, R2, and F2.

LIBRARY-{S / P}

Type of library entry you are modifying. Possible codes are:

Code	Meaning
S	Source statements (source library)
P	Procedures (source library)

WORK-code

Location of the simulation area containing space the program can use as a work area. Possible codes are R1, F1, R2, and F2.

RESER-{YES / NO / ONLY}

Specifies whether reserialization should be done when the entry is placed back in the source library. Possible information is:

Information	Meaning
YES	Reserialization is done.
NO	Reserialization is not done. NO is assumed if the RESER parameter is omitted.
ONLY	Reserialize only; no other maintenance is done. When this is coded, no REMOVE, REPLACE, INSERT, or CEND statements can be entered.

LIST-{YES / NO}

Specifies whether the source library entry should be listed as the modified entry is placed back in the source library. NO is assumed if the LIST parameter is omitted.

SEQFLD-xxyy

The starting and ending positions of the field that contains the sequence number. The sequence number can be up to 8 digits long. The starting position is entered first (xx) and then the ending position (yy). If this parameter is not entered, 9296 is assumed.

INCR-number

Increment value for sequence field if reserialization (RESER-YES or RESER-ONLY) is specified. The value can be up to 5 digits. If this parameter is not entered, a value of 10 is assumed.

Remove, Replace, Insert Parameters

FROM-seqno	The sequence number of the first statement to be used in the operation.
TO-seqno	The sequence number of the last statement to be used in the operation.
AFTER-seqno	The sequence number of the statement after which the new statements are to be added.

RENAME FUNCTION

Uses

- Change the name of a library entry.
- Change the name of library entries that have names beginning with certain characters.

Control Statement Summary

// RENAME FROM-code,LIBRARY-{S|P|O|R},NAME-name,NEWNAME-name

// RENAME FROM-code,LIBRARY-{S|P|O|R},NAME-characters.ALL,NEWNAME-characters

Considerations and Restrictions

- System modules should not be renamed on the active system simulation area (the simulation area the system was loaded from during IPL).
- Library maintenance modules should not be renamed on the active program simulation area.

Parameter Summary

FROM-code

Location of the simulation area that contains the entry you are renaming. Possible codes are R1, F1, R2, and F2.

LIBRARY-{S, P, O, R}

Type of library entry you are renaming. Possible codes are:

Code	Meaning
S	Source statements (source library)
P	Procedures (source library)
O	Object programs (object library)
R	Routines (object library)

NAME-name

Current name of the entry you are renaming. This is the name that identifies the entry in the library directory.

NAME-characters.ALL

Only those entries beginning with the indicated characters. (You can use up to 5 characters.)

NEWNAME-name

New name you want to give the entry. Follow these rules to construct the name:

- You can use any System/3 characters except blanks, commas, quotes, and periods. (Appendix A lists the characters.) However, the names of most IBM programs begin with a dollar sign ($). Therefore, to avoid possible duplication, do not use a dollar sign as the first character in the names you use for your entries. The first character must be alphabetic.
- You can use up to 6 characters, but you cannot use the names ALL, DIR, and SYSTEM. They have special meanings in the NAME parameter.

NEWNAME-characters

Beginning characters you want to use in names identifying the copies. (You can use up to 5 characters.)

OCL CONSIDERATIONS

The following OCL statements are needed to load the library maintenance program.

```
// LOAD $MAINT,code

// RUN
```

The code you supply depends on the location of the simulation area containing the library maintenance program. The codes are R1, F1, R2, and F2.

If the copy file-to-library or library-to-file function is used in this run of the $MAINT program, the necessary disk FILE OCL statements must be supplied. They must follow the LOAD statement and precede the RUN statement.

EXAMPLES

Figures 56 through 73 illustrate the functions of the library maintenance program. Figure 56 is an example of the OCL needed to load the utility program. The other figures are examples of the control statement necessary to carry out the specified function.

```
1   4   8   12  16  20  24  28  32  36
/&
// LOAD $MAINT,F1
// RUN
```

Explanation:

Library maintenance program is loaded from the simulation area F1 on drive 1.

Figure 56. OCL Load Sequence for Library Maintenance

```
1   4   8   12  16  20  24  28  32  36  40  44  48  52  56  60  64  68  72  76
// ALLOCATE TO-R1,SOURCE-12,OBJECT-45,SYSTEM-YES
// END
```

Explanation:

- Libraries are being created in the simulation area R1 on drive 1 (TO-R1 in ALLOCATE statement).
- Source library space is 12 tracks (SOURCE-12).
- Object library space is 45 tracks (OBJECT-45). The object library will contain system programs (SYSTEM-YES). Thus, the disk area also includes space for the scheduler work area.
- Directory size will be 3 tracks.

Figure 57. Allocate Example. Creating Both Source and Object Libraries on a Disk

```
// ALLOCATE TO-R1,SOURCE-15,WORK-F1
// END
```

Explanation:

- Source library is located in the simulation area R1 on drive 1 (TO-R1 in ALLOCATE statement).
- Size of the source library is being changed to 15 tracks (SOURCE-15).
- Any time the program changes the size of a library, it reorganizes the library. To do this, it needs a work area. This area is on the simulation area F1 on drive 1 (WORK-F1).

Figure 58. Allocate Example: Changing the Size of a Source Library

```
// ALLOCATE TO-R1,OBJECT-Ø
// END
```

Explanation:

- Object library is located in the simulation area R1 on drive 1 (TO-R1 in ALLOCATE statement).
- OBJECT-0 parameter tells the program to delete the object library. If a scheduler work area precedes the object library, it is also deleted.

Figure 59. Allocate Example: Deleting the Object Library from a Disk

```
// COPY FROM-F1,LIBRARY-O,NAME-SYSTEM,TO-R1
// END
```

Explanation:

- System programs are in the object library in the simulation area F1 on drive 1 (LIBRARY-O and FROM-F1 in COPY statement).
- The NAME parameter (NAME-SYSTEM) tells the program to copy the system programs.
- The disk that is to contain the copy is the simulation area R1 on drive 1 (TO-R1).

Figure 60. Copy Example: Copying Minimum System from One Disk to Another

```
1   4   8   12  16  20  24  28  32  36  40  44  48  52  56  60  64  68  72  76
// COPY FROM-R1,LIBRARY-ALL,NAME-DIR,TO-PRINT,OMIT-$.ALL
// END
```

Explanation:

- All library directories and the system directory in simulation area R1 on drive 1 are printed (COPY statement):

  FROM identifies the disk containing the directories.
  LIBRARY indicates which directories are to be printed.
  NAME and TO indicate that the program is to print directories.
  OMIT indicates that all entries beginning with a $ are not printed.

Figure 61. Copy Example: Printing Library Directories

```
1   4   8   12  16  20  24  28  32  36  40  44  48  52  56  60  64  68  72  76
// COPY FROM-R1,LIBRARY-O,NAME-ACCT,TO-F1,RETAIN-R
// END
```

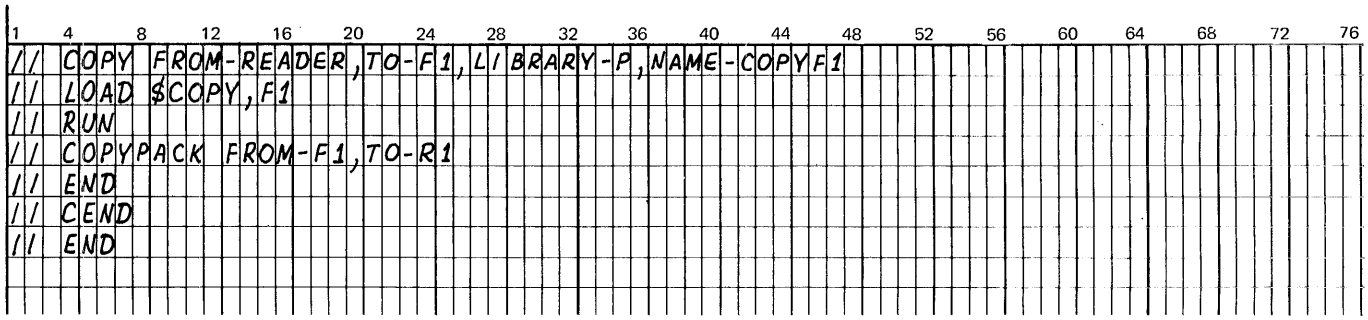
Explanation:

- LIBRARY-O, NAME-ACCT, and FROM-R1 in the COPY statement tell the program to read the object program named ACCT from the simulation area R1 on drive 1.

- TO-F1 tells the program to copy the object program to the simulation area F1 on drive 1. There is no NEWNAME parameter in the COPY statement. Therefore, the name the program uses in the simulation area F1 is ACCT (NAME-ACCT). Since the old version of the program already exists in the simulation area F1 under that name, the old version is replaced.

- The library maintenance program normally halts before replacing a library entry. The RETAIN-R parameter, however, tells the program to omit that halt.

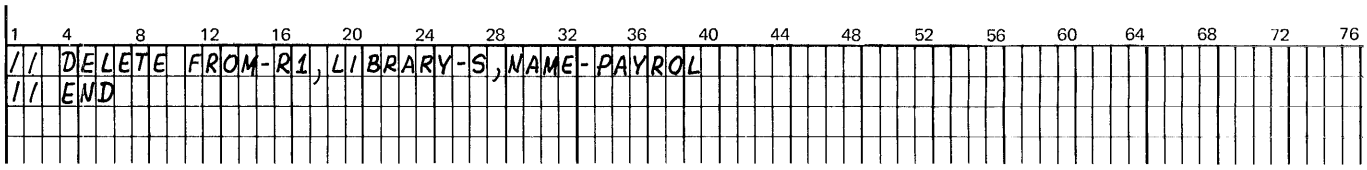
Figure 62. Copy Example: Copying Object Program to F1



Explanation:

- FROM-READER tells the library maintenance program to read the statements from the system input device.
- To procedure (LIBRARY-P) is written to the source library on F1 (TO-F1), named COPYF1 (NAME-COPYF1), and given the default attribute of temporary.
- All statements following the // COPY statement are entered into the library until the // CEND statement is read to terminate the COPY.
- // END following the // CEND statement is optional here. If used it terminates the library maintenance program. If it is not used, more control statements may be entered following the // CEND statement.

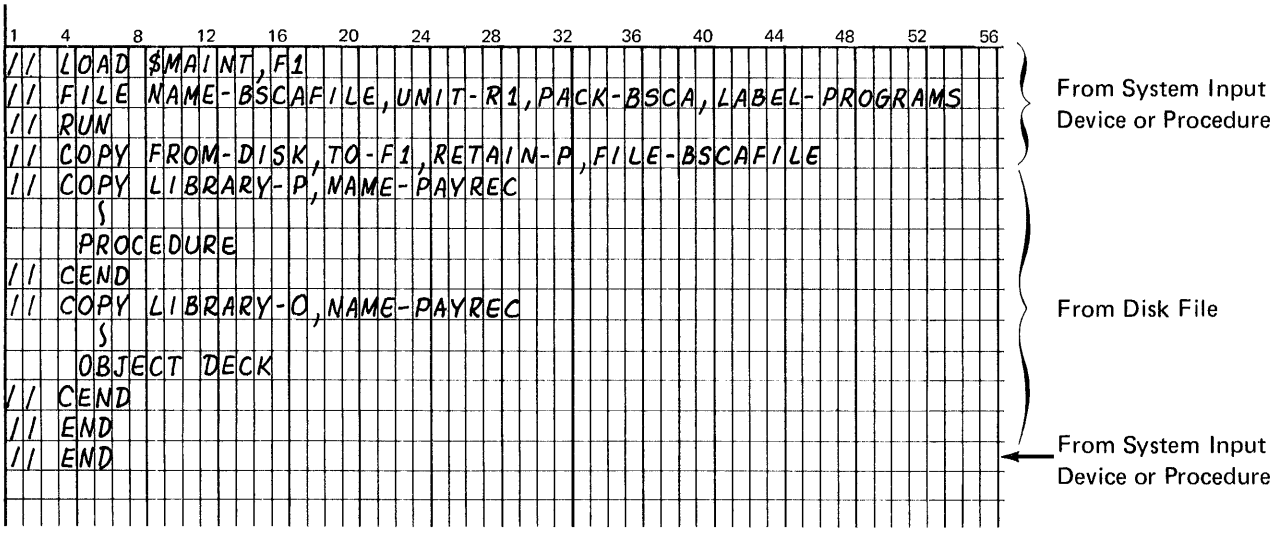
Figure 63. Copy Example: Copying Procedure from System Input Device



Explanation:

The program deletes a set of source statements (LIBRARY-S in DELETE statement) named PAYROL (NAME-PAYROL) from the simulation area R1 on drive 1 (FROM-R1) that has a temporary attribute.

Figure 64. Delete Example: Deleting an Entry from a Library



Explanation:

- The OCL for a file-to-library copy must contain a FILE statement for the disk file.
- The filename on the // COPY statement (FILE-BSCAFILE) matches the filename on the OCL FILE statement (NAME-BSCAFILE).
- The // COPY statement does not contain a RECL parameter, so a record length of 96 is assumed.
- All source and object decks in the disk file must have a // COPY statement as the first card image and a // CEND statement as the last card image to indicate the end of the copy for each deck. These // statements (including the // END statement) are logged with XX replacing the // to indicate they were read from disk rather than from the system input device or a procedure.

  Note: The // CEND statement is not printed.
- The // END statement read from the file (printed XX END), causes the next statement to be read from the system input device or procedure. A // END statement must still be read from the system input device or procedure to indicate the end of the library maintenance control statements.

  Note: The // END statement in the file is optional because the system recognizes the physical end of the data file and terminates the copy.

Figure 65. Copy Example: Disk File to Library

```
// LOAD $MAINT,F1
// FILE NAME-BACKUP,UNIT-D1,PACK-D1D1D1,LOCATION-20/0,TRACKS-80
// RUN
// COPY FROM-R1,TO-DISK,RECL-80,FILE-BACKUP
// ENTRY LIBRARY-ALL,NAME-PAY.ALL
// ENTRY LIBRARY-S,NAME-ALL
// ENTRY LIBRARY-O,NAME-INVENT
// NEND
// END
```

Explanation:

- The OCL for a library-to-file copy must contain a FILE statement for the disk file.
- The filename on the // COPY statement (FILE-BACKUP) matches the filename on the OCL FILE statement (NAME-BACKUP).
- A sequential file with record length of 80 (RECL-80) is created on D1.
- The file will contain entries from all libraries with names beginning with the characters PAY, all source library entries, and object entry INVENT.
- The copy to file BACKUP is terminated by the // NEND statement.
- The // END statement following the // NEND is required. It terminates the library maintenance program.

Figure 66. Copy Example: Library-to-Disk File

```
// DELETE FROM-R1,LIBRARY-ALL,NAME-INV.ALL
// END
```

Explanation:

- The entries being deleted are in the simulation area R1 on drive 1 (FROM-R1 in DELETE statement).
- The program deletes all entries from both source and object libraries (LIBRARY-ALL) that have names beginning with the characters INV (NAME-INV.ALL), with temporary attributes.

Figure 67. Delete Example: Deleting All Entries with Names that Begin with Certain Characters

```
// DELETE FROM-R1,LIBRARY-P,NAME-ALL,RETAIN-T
// END
```

Explanation:

- The entries being deleted are in the simulation area R1 on drive 1 (FROM-R1 in DELETE statement).
- All temporary procedures are being deleted from the source library (LIBRARY-P,NAME-ALL).

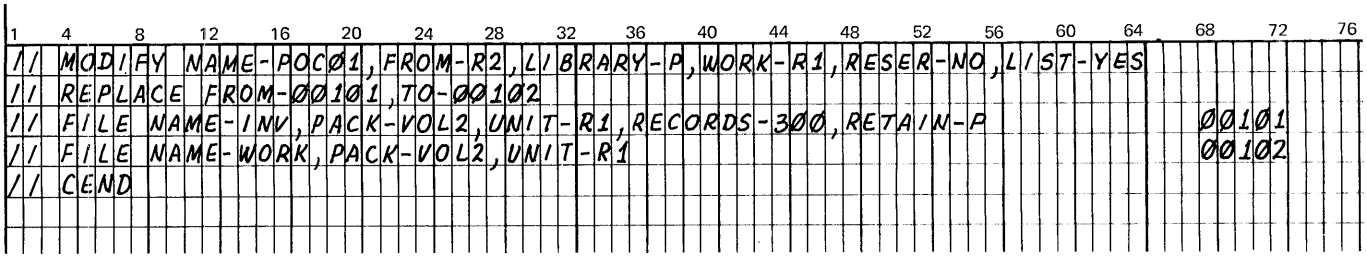
Figure 68. Delete Example: Deleting All Library Entries of One Type

```
// MODIFY NAME-INPUT1,FROM-R1,LIBRARY-S,WORK-R1,RESER-YES,LIST-NO,
//        SEQFLD-0105,INCR-1
// REMOVE FROM-00124,TO-00156
// CEND
```

Explanation:

- The source module named INPUT1 in simulation area R1 on drive 1 is being modified (NAME-INPUT1,FROM-R1, LIBRARY-S in the MODIFY statement).
- The work space is on R1 (WORK-R1).
- The sequence numbers are in positions 1-5 of the statements (SEQFLD-0105).
- Sequence numbers 00124-00156 are being deleted from the module (FROM-00124,TO-00156 in the REMOVE statement).
- The module is reserialized with increments of one (RESER-YES,INCR-1).
- The module is not listed (LIST-NO).

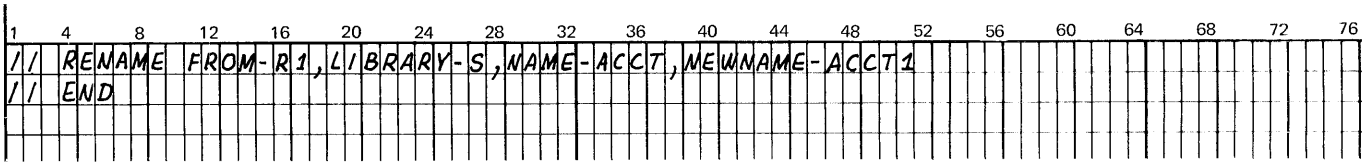
Figure 69. Modify Example: Removing Source Statements from a Module



Explanation:

- The procedure named POC01 in simulation area R2 on drive 2 is being modified (NAME-POC01,FROM-R2,LIBRARY-P in the MODIFY statement).
- The work space is on R1 (WORK-R1).
- The sequence numbers are in default positions 92 through 96.
- Statements with sequence numbers 00101 and 00102 are being replaced (FROM-00101,TO-00102 in the REPLACE statement).
- The module is not reserialized (RESER-NO).
- The module is listed (LIST-YES).

Figure 70. Modify Example: Replacing Statements in a Procedure



Explanation:

- The simulation area R1 on drive 1 contains the entry being renamed (FROM-R1 in RENAME statement).
- The entry is a set of source statements in the source library (LIBRARY-S). Its name is ACCT (NAME-ACCT).
- The entry name is being changed to ACCT1 (NEWNAME-ACCT1).

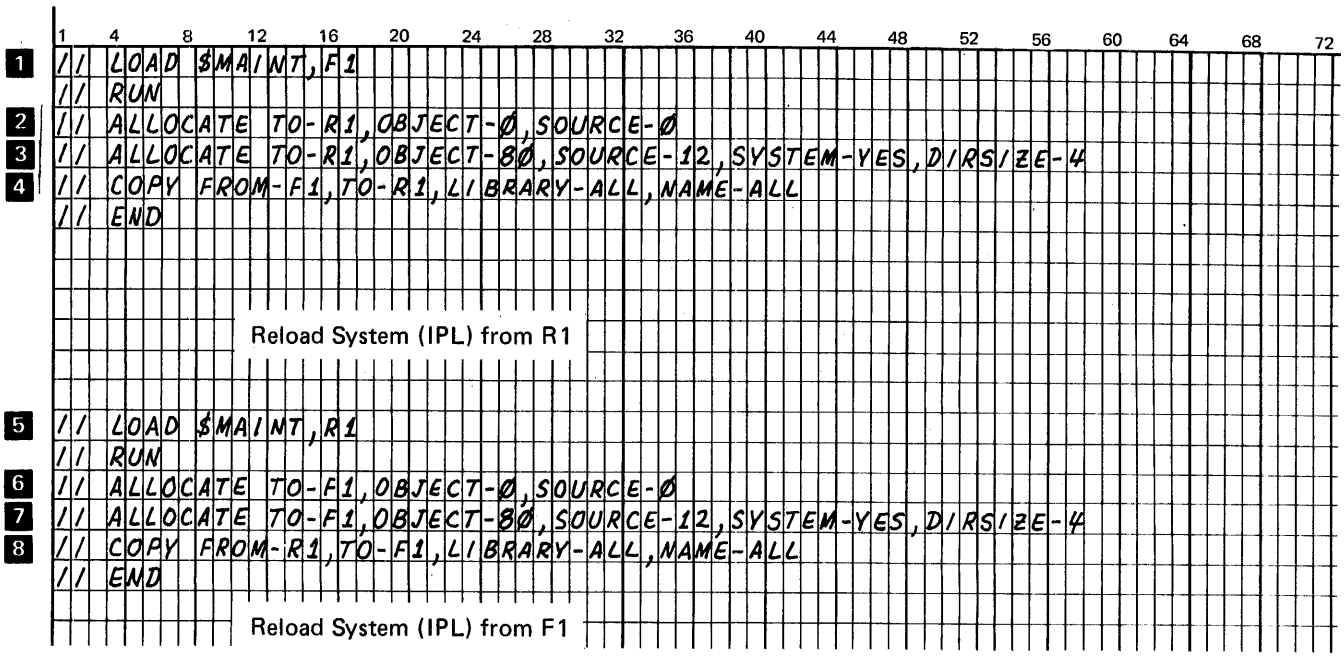
Figure 71. Rename Example: Renaming a Set of Source Statements in a Source Library

```
1    4    8    12   16   20   24   28   32   36   40   44   48   52   56   60   64   68   72   76
// MODIFY FROM-F1,WORK-F1,NAME-COST,LIBRARY-S,
//        RESER-YES,SEQFLD-8084,LIST-YES
// INSERT AFTER-00070
   00080I                                        3    8 DATE
// END
```

Explanation:

- The source module COST in simulation area F1 on drive 1 is being modified (FROM-F1,NAME-COST,LIBRARY-S in the MODIFY statement).
- The work space is on F1 (WORK-F1).
- The sequence numbers are in position 80 through 84 of the statements (SEQFLD-8084).
- A statement is being inserted after statement number 00070 (AFTER-00070 in the INSERT statement).
- The module is reserialized with the default increment value of 10 (RESER-YES).
- The module is listed (LIST-YES).

Figure 72. Modify Example: Inserting a Statement in a Source Module



Explanation:

1. The system and $MAINT are both loaded from F1.
2. The libraries on R1 are deallocated (if present).
3. New library space is allocated on R1.
4. The libraries are copied from F1 to R1. The object library is reorganized as it is copied. Temporary entries become permanent when copied.
5. The system and $MAINT are now loaded from R1.
6. The libraries on F1 are deallocated.
7. New library space is allocated on F1.
8. The libraries are copied back to F1. The simulation area R1 could be used as a backup pack. It contains the same libraries as F1.

Figure 73. Reorganizing the System Simulation Area

REASSIGN ALTERNATE TRACK PROGRAM-$RSALT

When it is necessary to transport a 3340 data module from System/3 to System/360 or System/370, you must run the reassign alternate track program ($RSALT) before you run the DOS/OS initialization program.

On a 3340 data module initialized on System/3, there are 40 alternate tracks on cylinders 167 and 168. On a System/360 or System/370 3340 data module, there are 24 alternate tracks from cylinders 167 and 168 to cylinders 208 and 209. Consequently, if a 3340 data module initialized on System/3 has more than 24 defective primary tracks, it cannot be initialized by System/360 or System/370.

Note: Data interchange is not supported between the System/3 and the System/360 or System/370, so this program cannot be used for that purpose. System/3 data existing on the data module before $RSALT is run will be lost.

Control Statement Summary

```
// ALTA UNIT-D2,PACK-name

// END
```

Parameter Summary

UNIT-D2	Specifies the location of the data module that you want to modify.
PACK-name	Specifies the name of the data module you want to modify.

Parameter Descriptions

UNIT Parameter

The UNIT parameter (UNIT-D2) specifies the location of the data module that you want to modify. The program can modify only data module D2 during a program run. $RSALT cannot be run on D2 if the simulation areas (R2 and F2) are active. The OCL statement, // SIMULATE OFF, must be used before $RSALT is executed.

PACK Parameter

The PACK parameter (PACK-name) tells the program the name of the data module to be modified. The parameter length must not exceed 6 characters. It can contain any of the standard System/3 characters except apostrophes, commas, or leading or embedded blanks.

The reassign alternate track program compares the name in the PACK parameter with the name on the data module to ensure that they match. If the names do not match, the program halts with an error message. In this way, the program ensures that it is using the right data module.

OCL Considerations

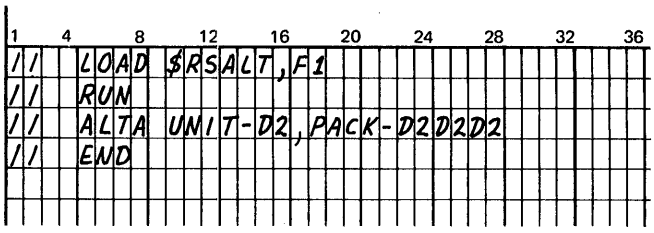
The following OCL statements are needed to load the reassign alternate track program:

```
// LOAD $RSALT,code

// RUN
```

The code you supply depends on the location of the simulation area containing the reassign alternate track program. The codes are R1 and F1.

Example

The following illustration shows an example of the control statements required to execute the $RSALT program:



Explanation:

The 3340 data module on drive 2 is to be modified to System/360–System/370 format.

Recover Index Program—$RINDX

The Recover Index ($RINDX) program is used to recover the records added to an indexed file if, for any reason, the program adding the records is terminated before end of job.

The Recover Index program should be:

- Executed as soon as possible after the abnormal termination, and
- Executed in a dedicated system

Each indexed file for which records are to be recovered must be described by an OCL FILE statement. The description must include the filename, unit code, and pack ID. You may also include OCL FILE statements for other than indexed files; however, the Recover Index program will not attempt to recover records in other file organizations. The following example shows a FILE statement for each file to be checked for record recovery:

```
1   4       8       12      16      20      24      28      32      36      40      44      48      52
// LOAD $RINDX,R2
// FILE NAME-$INDEX45,UNIT-D2,PACK-D2D2D2,TRACKS-30
// FILE NAME-CONSEC,UNIT-D1,PACK-D1D1D1
// FILE NAME-DIRECT,UNIT-D1,PACK-D1D1D1
// FILE NAME-IN45Ø1,UNIT-D1,PACK-D1D1D1
// FILE NAME-IN45Ø2,UNIT-D1,PACK-D1D1D1
// FILE NAME-IN45Ø3,UNIT-D1,PACK-D1D1D1
// FILE NAME-IN45Ø4,UNIT-D1,PACK-D1D1D1
// FILE NAME-IN45Ø5,UNIT-D1,PACK-D1D1D1
// FILE NAME-IN45Ø6,UNIT-D1,PACK-D1D1D1
// FILE NAME-IN45Ø7,UNIT-D1,PACK-D1D1D1,TRACKS-10
// RUN
```

The $INDEX45 file is a work file used to decrease the processing time for sorting the indexes of large indexed files.

The functions of the Recover Index program for each file organization are:

- *Indexed File:*
  - If added keys exist for the file when the abnormal termination occurs, $RINDX updates the end-of-index and end-of-data pointers. File information—defined as file label, file type, pack label, and file date—is printed. The last added key for this file is also printed.
  - If keys had not been added when the abnormal termination occurred, only the file information is printed.
- If a *consecutive* file is detected, only the file information is printed.
- If a *direct* file is detected, only the file information is printed.
- If the Recover Index program cannot find the file described by the OCL FILE statement, the file information and the message FILE NOT AVAILABLE is printed.

The following printout is a result of processing each FILE statement shown in the previous example:

```
$RINDX-  FILE RECOVERY PROGRAM    DATE-XX/XX/XX

FILE     FILE     PACK     FILE     LAST ADD KEY
LABEL    TYPE     LABEL    DATE         INCLUDED

CONSEC     C      D1D1D1   021976
DIRECT     D      D1D1D1   021976
IN4501     I      D1D1D1   021976   00971
IN4502     I      D1D1D1   021976   002031
IN4503     I      D1D1D1   021976   0004131
IN4504     I      D1D1D1   021976   00005191
IN4505     I      D1D1D1   021976   000007211
IN4506     I      D1D1D1   021976   0000007211
IN4507            D1D1D1   021676   FILE NOT AVAILABLE

ALL FILES PROCESSED
```

After all OCL FILE statements have been processed, an ALL FILES PROCESSED message is printed. The index is then sorted and the VTOC (volume table of contents) updated.

Note: After the ALL FILES PROCESSED message is printed, do not cancel or start the next job prior to actual end of job. Processing continues with sorting the index and updating the VTOC.

OCL CONSIDERATIONS

The following OCL statements are needed to load and execute the Recover Index program:

```
// LOAD $RINDX,code

// FILE NAME-xxxxxxxx,UNIT-xx,PACK-xxxxxx

// RUN
```

The code you supply depends on the location of the disk containing the Recover Index program. Possible codes are R1, F1, R2, F2.

Considerations and Restrictions

If a disk I/O error occurs during the execution of $RINDX, the file information and error message DISK I/O ERROR is printed. A halt then occurs; options are:

- Continue processing with the next file
- Cancel the job

If halt DDØP (keysort duplicate key) occurs during the execution of $RINDX, it may indicate that the program was abnormally terminated during the process of sorting the index. Continue processing until end of job for $RINDX. If the file is not known to have duplicate keys, use the Copy/Dump program ($COPY) with REORG-NO and an OMIT or DELETE parameter to rebuild the index.

EXAMPLES

In the following example, the Recover Index program is loaded from R2. The printout shows that keys were added to each of the files except IN4403 before the abnormal termination.

```
//  LOG PRINTER
//  LOAD $RINDX,R2
//  FILE NAME-$INDEX45,UNIT-D2,PACK-D2D2D2,TRACKS-30
//  FILE NAME-$INDEX44,UNIT-R2,PACK-R2R2R2,TRACKS-20
//  FILE NAME-IN4501,UNIT-D1,PACK-D1D1D1
//  FILE NAME-IN4502,UNIT-D1,PACK-D1D1D1
//  FILE NAME-IN4503,UNIT-D1,PACK-D1D1D1
//  FILE NAME-IN4401,UNIT-R1,PACK-R1R1R1
//  FILE NAME-IN4402,UNIT-R1,PACK-R1R1R1
//  FILE NAME-IN4403,UNIT-R1,PACK-R1R1R1
//  RUN

$RINDX -     FILE RECOVERY PROGRAM    DATE-XX/XX/XX

FILE     FILE     PACK     FILE     LAST ADD KEY
LABEL    TYPE     LABEL    DATE          INCLUDED

IN4401     I      R1R1R1   022676   000000000000000000000971
IN4402     I      R1R1R1   022676   000000000000000000002031
IN4403     I      R1R1R1   022676
IN4501     I      D1D1D1   022676   00971
IN4502     I      D1D1D1   022676   002031
IN4503     I      D1D1D1   022676   0004131

ALL FILES PROCESSED
  1 DD KS     1                     $RINDX01
BEGIN KEY SORT/MERGE - IN4401
  1 DD KS     1                     $RINDX01
BEGIN KEY SORT/MERGE - IN4402
  1 DD KS     1                     $RINDX01
BEGIN KEY SORT/MERGE - IN4403
  1 DD KS     1                     $RINDX01
BEGIN KEY SORT/MERGE - IN4501
  1 DD KS     1                     $RINDX01
BEGIN KEY SORT/MERGE - IN4502
  1 DD KS     1                     $RINDX01
BEGIN KEY SORT/MERGE - IN4503




  1 CT EJ     1                     $RINDX01
02/26/76 00.00.19 00.02.47
```

Character	Hexadecimal Equivalent
Blank	40
¢	4A
.	4B
<	4C
(	4D
+	4E
|	4F
&	50
!	5A
$	5B
*	5C
)	5D
;	5E
¬	5F
- (minus)	60
/	61
,	6B
%	6C
_ (underscore)	6D
>	6E
?	6F
:	7A

Character	Hexadecimal Equivalent
#	7B
@	7C
' (apostrophe)	7D
=	7E
"	7F
A	C1
B	C2
C	C3
D	C4
E	C5
F	C6
G	C7
H	C8
I	C9
}	D0
J	D1
K	D2
L	D3
M	D4
N	D5
O	D6
P	D7

Character	Hexadecimal Equivalent
Q	D8
R	D9
S	E2
T	E3
U	E4
V	E5
W	E6
X	E7
Y	E8
Z	E9
0	F0
1	F1
2	F2
3	F3
4	F4
5	F5
6	F6
7	F7
8	F8
9	F9

READER'S COMMENT FORM

IBM System/3 Model 12
System Control Programming
Reference Manual

GC21-5130-0

YOUR COMMENTS, PLEASE . . .

Your comments assist us in improving the usefulness of our publications; they are an important part of the input used in preparing updates to the publications. All comments and suggestions become the property of IBM.

Please do not use this form for technical questions about the system or for requests for additional publications; this only delays the response. Instead, direct your inquiries or requests to your IBM representative or to the IBM branch office serving your locality.

Corrections or clarifications needed:

Page	*Comment*

I would like a reply. ☐

Name _____

Address _____

- Thank you for your cooperation. No postage necessary if mailed in the U.S.A.

Cut Along Line

Fold

Fold

FIRST CLASS
PERMIT NO. 387
ROCHESTER, MINN.

BUSINESS REPLY MAIL

NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY . . .

IBM Corporation
General Systems Division
Development Laboratory
Publications, Dept. 245
Rochester, Minnesota 55901

Fold

Fold

S/3 Model 12 System Control Programming Reference (File No. S3-36) Printed in U.S.A. GC21-5130-0

International Business Machines Corporation
General Systems Division
5775D Glenridge Drive N.E.
Atlanta, Georgia 30301
(USA Only)

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

GC21-5130-0

S/3 Model 12 System Control Programming Reference (File No. S3-36) Printed in U.S.A. GC21-5130-0

International Business Machines Corporation
General Systems Division
5775D Glenridge Drive N.E.
Atlanta, Georgia 30301
(USA Only)

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