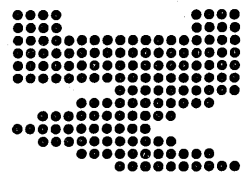
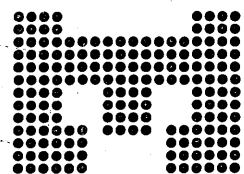
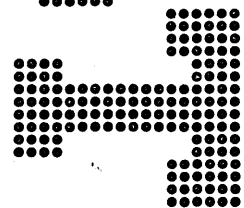
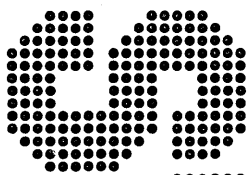
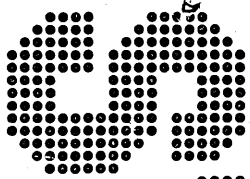


**IBM System/3  
Model 6  
Operation Control Language and  
Disk Utility Programs  
Reference Manual**

**Program Number 5703-SC1**



## PREFACE

This publication is intended for use by programmers who are doing either of the following:

1. Writing Operation Control Language (OCL) statements needed to run programs in the system.
2. Writing utility control statements necessary to run disk utility programs supplied by the system.

### Prerequisite Publications

*IBM System/3 Model 6 Introduction, GA21-9122*

### Other Publications Referenced in This Manual

*IBM System/3 Model 6 Operator's Guide, GC21-7501*

*IBM System/3 Disk Sort Reference Manual, SC21-7522*

*IBM System/3 Model 6 Conversation Utility Programs Reference Manual, SC21-7528*

*IBM System/3 Model 6 Utility Program for IBM 1255 Magnetic Character Reader Reference Manual, SC21-7527*

*IBM System/3 Model 6 RPG II Reference Manual, SC21-7517*

### Third Edition (September 1971)

This is a major revision of, and obsoletes, GC21-7516-1 and Technical Newsletter GN21-7575. Changes are indicated by a vertical line to the left of the change.

This edition applies to version 5, modification 0 of the IBM System/3 Model 6 and to all subsequent versions and modifications until otherwise indicated in new editions or Technical Newsletters. Changes are continually made to the specifications herein; before using this publication in connection with the operation of IBM Systems, consult the latest IBM System/3 Newsletter Order Number GN20-2228 for the editions that are applicable and current.

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

A form for reader's comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Programming Publications, Department 425, Rochester, Minnesota 55901.

## MACHINE REQUIREMENTS

Conversational OCL and all utility programs except Library Maintenance can be done using the minimum configuration of System/3 Model 6.

The minimum configuration is as follows:

- IBM 5406 Processing Unit, Model B2 — including keyboard
- IBM 5444 Disk Storage Drive
- IBM 5213 Printer

OCL statements from cards and Library Maintenance functions involving cards require an additional unit: IBM 5496 Data Recorder, Model 1 with System/3 Model 6 On-Line Feature.

# CONTENTS

MACHINE REQUIREMENTS . . . . .	i	DATE (File Date) . . . . .	52
HOW TO USE THIS MANUAL . . . . .	1	Restriction During File Creation . . . . .	52
PART I. OPERATION CONTROL LANGUAGE . . . . .	3	HALT . . . . .	52
HOW TO USE PART I . . . . .	5	LOAD NAME . . . . .	53
SUMMARY OF CONVERSATIONAL OCL . . . . .	7	For Customer Programs . . . . .	53
The Job Cycle . . . . .	7	For System Programs . . . . .	53
The Four OCL Cycles . . . . .	9	MODIFY . . . . .	54
BUILD and CALL Cycles . . . . .	9	MODIFY; Changing a Previous OCL Statement . . . . .	56
System-Operator Interaction During Keyword Prompting . . . . .	11	MODIFY; Deleting a Previous OCL Statement . . . . .	57
End-of-Statement Keys . . . . .	13	MODIFY; Entering Comments . . . . .	58
Program Start (PROG START) or Enter Plus (ENTER+) . . . . .	13	MODIFY; Cancelling Job . . . . .	59
Enter Minus (ENTER-) . . . . .	13	MODIFY; Entering Forms . . . . .	60
Statement Numbers in an OCL Cycle . . . . .	13	MODIFY; Including Control Statements . . . . .	62
Comments . . . . .	13	NOHALT . . . . .	63
Inquiry Interrupt . . . . .	14	READY . . . . .	64
Restrictions During Inquiry . . . . .	14	RUN . . . . .	64
Keyword Sequence for OCL Load Cycle . . . . .	15	SWITCH . . . . .	64
Keyword-Response Summary (Load Cycle) . . . . .	16	Indicator Settings . . . . .	64
Keyword Sequence for OCL Build Cycle . . . . .	26	IPL Considerations . . . . .	64
Keyword-Response Summary (Build Cycle) . . . . .	27	Duration of SWITCH Setting . . . . .	64
Keyword Sequence for OCL Call Cycle . . . . .	41	Operator-System Interaction for SWITCH Statement (LOAD Cycle) . . . . .	65
Keyword-Response Summary (Call Cycle) . . . . .	42	Operator-System Interaction for SWITCH Statement (BUILD Cycle) . . . . .	66
KEYWORD DESCRIPTIONS . . . . .	45	Operator-System Interaction for SWITCH Statement (CALL Cycle) . . . . .	67
BUILD NAME . . . . .	45	SAMPLE JOBS . . . . .	69
Duplicate Procedure Names . . . . .	45	Sample Job 1. Initialize Disk . . . . .	70
Deleting a Source Library Procedure . . . . .	45	Explanation . . . . .	71
BUILDC NAME . . . . .	45	Sample Job 2. Compile an RPG Source Program . . . . .	72
CALL NAME . . . . .	45	Explanation . . . . .	73
COMPILE OBJECT . . . . .	46	Sample Job 3. Process Customer Program "INVUPD" . . . . .	74
SOURCE . . . . .	46	Explanation . . . . .	75
In a LOAD Cycle . . . . .	46	Sample Job 4. Copy File Disk to Disk . . . . .	76
In a BUILD Cycle . . . . .	46	Explanation . . . . .	77
UNIT; Source Unit . . . . .	46	Sample Job 5. Multi-File BUILD . . . . .	78
DATE (System Date) . . . . .	47	Explanation . . . . .	80
Overriding the System Date . . . . .	47	Sample Job 6. Multi-File CALL . . . . .	81
Format of the DATE Statement . . . . .	47	Explanation . . . . .	82
FILE NAME . . . . .	48	PART II. DISK UTILITY PROGRAMS . . . . .	83
FILE NAME for Customer Programs . . . . .	48	HOW TO USE PART II . . . . .	85
FILE NAME for \$RPG, \$DSORT, \$COPY, \$MICR, and \$KDE . . . . .	48	Writing Utility Control Statements . . . . .	85
System-Operator Interaction During Prompting of File Keywords . . . . .	49	Writing OCL Statements . . . . .	85
Multiple Files . . . . .	50	Capital Letters, Numbers, and Special Characters . . . . .	85
UNIT; File Unit . . . . .	50	INTRODUCTION . . . . .	87
PACK . . . . .	50	General Program Operation . . . . .	87
LABEL . . . . .	50	Library Maintenance Program . . . . .	88
RECORDS (and TRACKS) . . . . .	50	Control Statements . . . . .	88
Responding to TRACKS . . . . .	51	DISK INITIALIZATION PROGRAM . . . . .	91
Responding to RECORDS . . . . .	51	Control Statement Summary for \$INIT . . . . .	92
LOCATION . . . . .	51	Parameter Summary . . . . .	93
RETAIN . . . . .	51	Parameter Descriptions . . . . .	94
File Creation . . . . .	51	TYPE Parameter (UIN) . . . . .	94
Changing File Designation of Existing File . . . . .	52	UNIT Parameter (UIN) . . . . .	94
Deleting Files . . . . .	52		



VERIFY Parameter (UIN) . . . . .	95	DISK COPY/DUMP PROGRAM . . . . .	131
ERASE Parameter (UIN) . . . . .	95	Control Statement Summary for \$COPY . . . . .	132
CAP Parameter . . . . .	95	Parameter Summary . . . . .	134
PACK Parameter (VOL) . . . . .	96	Parameter Descriptions . . . . .	136
ID (Identification) Parameter (VOL) . . . . .	96	FROM and TO Parameters (COPYPACK) . . . . .	136
OCL Considerations . . . . .	97	OUTPUT Parameter (COPYFILE) . . . . .	136
Example . . . . .	98	DELETE Parameter (COPYFILE) . . . . .	137
Primary Initialization of Two Disks . . . . .	98	REORG (Reorganize) Parameter (COPYFILE) . . . . .	137
Messages for Disk Initialization . . . . .	99	WORK Parameter (COPYFILE) . . . . .	137
		SELECT KEY and PKY Parameters (SELECT) . . . . .	138
ALTERNATE TRACK ASSIGNMENT PROGRAM . . . . .	101	SELECT RECORD Parameters (SELECT) . . . . .	138
Control Statement Summary for \$ALT . . . . .	102	Copying Multi-Volume Files . . . . .	139
Parameter Summary . . . . .	103	Maintaining Proper Volume Sequence Numbers . . . . .	139
Parameter Descriptions . . . . .	104	Maintaining Correct Relative Record Numbers . . . . .	139
PACK Parameter . . . . .	104	Direct File Attributes . . . . .	139
UNIT Parameter . . . . .	104	Copying Multi-Volume Index Files . . . . .	139
VERIFY Parameter . . . . .	104	OCL Considerations . . . . .	140
ASSIGN Parameter . . . . .	105	Examples . . . . .	143
UNASSIGN Parameter . . . . .	105	Copying an Entire Disk . . . . .	143
OCL Considerations . . . . .	106	Copying a File From One Disk to Another . . . . .	144
Example . . . . .	107	Printing Part of a File . . . . .	146
Conditional Assignment . . . . .	107		
Messages for Alternate Truck Assignment . . . . .	108	LIBRARY MAINTENANCE PROGRAM . . . . .	147
		Library Description . . . . .	147
ALTERNATE TRACK REBUILD PROGRAM . . . . .	109	Organization of This Section . . . . .	148
Control Statement Summary for \$BUILD . . . . .	109	ALLOCATE . . . . .	149
Parameter and Substitute Data Summary . . . . .	110	Uses . . . . .	149
Parameter and Substitute Data Descriptions . . . . .	111	Control Statement Summary . . . . .	149
PACK Parameter . . . . .	111	Parameter Summary . . . . .	150
UNIT Parameter . . . . .	111	Parameter Descriptions . . . . .	151
TRACK Parameter . . . . .	111	COPY . . . . .	156
LENGTH Parameter . . . . .	111	Uses . . . . .	156
DISP (Displacement) Parameter . . . . .	111	Control Statement Summary: Reader-To-Disk . . . . .	158
Substitute Data . . . . .	111	Control Statement Summary: Disk-To-Printer . . . . .	160
OCL Considerations . . . . .	112	Control Statement Summary: Disk-To-Card . . . . .	161
Example . . . . .	113	Control Statement Summary: Disk-To-Printer and	
Correcting Characters on an Alternate Track . . . . .	113	Card . . . . .	162
		Parameter Summary . . . . .	163
FILE AND VOLUME LABEL DISPLAY PROGRAM . . . . .	115	Parameter Descriptions . . . . .	166
Control Statement Summary for \$LABEL . . . . .	115	DELETE . . . . .	176
Parameter Summary . . . . .	116	Uses . . . . .	176
Parameter Descriptions . . . . .	116	Control Statement Summary . . . . .	177
UNIT Parameter . . . . .	116	Parameter Summary . . . . .	178
LABEL Parameter . . . . .	116	RENAME . . . . .	179
OCL Considerations . . . . .	119	Use . . . . .	179
Example . . . . .	120	Control Statement Summary . . . . .	179
Printing VTOC Information for Two Files . . . . .	120	Parameter Summary . . . . .	180
		OCL Considerations . . . . .	181
FILE DELETE PROGRAM . . . . .	121	ALLOCATE Examples . . . . .	182
Control Statement Summary for \$DELET . . . . .	122	Creating Both Source and Object Libraries on a Disk . . . . .	182
Parameter Summary . . . . .	124	Changing the Size of a Source Library . . . . .	183
Parameter Descriptions . . . . .	126	Deleting the Object Library from a Disk . . . . .	184
PACK Parameter . . . . .	126	COPY Examples . . . . .	185
UNIT Parameter . . . . .	126	Copying Minimum System from One Disk to Another . . . . .	185
LABEL Parameter . . . . .	126	Printing Library Directories . . . . .	186
DATE Parameter . . . . .	126	Replacing a Library Entry: Replacement Coming from	
DATA Parameter (Remove Only) . . . . .	126	Another Disk . . . . .	187
OCL Considerations . . . . .	127		
Example . . . . .	128		
Deleting One of Several Files Having the Same			
Name . . . . .	128		
Removing One File . . . . .	129		

DELETE Examples . . . . .	188
Deleting a Temporary Entry from a Library . . . . .	188
Deleting All Temporary Entries With Names That Begin With Certain Characters . . . . .	189
Deleting All Permanent Library Entries of One Type . . . . .	190
RENAME Example . . . . .	191
Renaming a Set of Source Statements in a Source Library . . . . .	191
APPENDIX A: ADVANCED TOPICS FOR OCL . . . . .	193
Multi-Volume Files . . . . .	193
File Statements for Multi-Volume Files . . . . .	193
OCL Considerations . . . . .	194
List Requirements . . . . .	194
OCL Considerations for Multi-Volume Files . . . . .	196
Coding Multi-Volume File Statements . . . . .	198
Changing Multi-Volume File Statements with Modify Keyword . . . . .	198
Sample Job 7. Updating Multi-Volume Master File . . . . .	198
Explanation . . . . .	199
Sample Job 8. Creating a Multi-Volume Indexed File . . . . .	199
Explanation . . . . .	200
Sample Job 9. Maintaining a Multi-Volume Indexed File with Packed Keys. . . . .	201
Including Sort Source or Utility Control Statements in a Procedure . . . . .	202
Sample Job 10. Including Utility Control Statements in a Procedure . . . . .	202
Increasing File Size of the RPG Procedure . . . . .	203
Entering RPG II Source Statements from the Keyboard at Compile Time . . . . .	203
Chained Procedures . . . . .	204
Sample Job 11. Chained Procedure . . . . .	209
APPENDIX B: RECORDS – TRACKS CONVERSION . . . . .	212
For Sequential or Direct Files . . . . .	212
For Indexed Files . . . . .	212
APPENDIX C: DISK ORGANIZATION . . . . .	213
Volume Table of Contents (VTOC) . . . . .	213
Source Library . . . . .	213
Object Library . . . . .	213
Files . . . . .	213

APPENDIX D: OCL FOR THE 22" PRINTER (IBM 2222 PRINTER) . . . . .	214
Using the FORMS Statement with the 22" Printer . . . . .	214
Log Device . . . . .	214
MODIFY – Entering the Keyword FORMS . . . . .	215
APPENDIX E: OCL FOR THE IBM 2265-2 DISPLAY . . . . .	216
READY – Entering LOG . . . . .	217
MODIFY – Entering LOG . . . . .	218
APPENDIX F: OPERATOR'S OCL GUIDE . . . . .	219
APPENDIX G: CARD OCL FOR MODEL 6 . . . . .	220
Assigning Data Recorder As System Input Device . . . . .	220
Returning Control to Keyboard . . . . .	220
Card Format of OCL Statements . . . . .	220
General Coding Rules . . . . .	223
Statement Order . . . . .	223
Coding Multi-Volume File Parameters . . . . .	224
APPENDIX H: OCL ERROR MESSAGES . . . . .	225
APPENDIX I: CO-RESIDENT SYSTEMS . . . . .	231
APPENDIX J: IBM SYSTEM/3 STANDARD CHARACTER SET . . . . .	232
GLOSSARY . . . . .	233
INDEX . . . . .	235



## HOW TO USE THIS MANUAL

This publication contains two parts. Part I describes Operation Control Language (OCL) statements. Part II describes disk utility programs.


### Part I

Refer to Part I if you want to know:

1. What an OCL statement is.
2. How to enter the OCL statements required to run your jobs.

### Part II

Refer to Part II if you want to know:

1. What disk utility programs are supplied with the system.
  2. The function of each disk utility program.
  3. The Operation Control Language (OCL) statements and utility control statements necessary to request each disk utility program.
- 





**PART I.  
OPERATION CONTROL LANGUAGE**



.

Part I of this manual is designed to help you fill out the OCL guide sheets your operator will use in response to the OCL prompting for each job. You can either design an operator's OCL guide sheet for your installation or use the pre-printed form that is available (see *Appendix F: Operator's OCL Guide*).

This part contains a main section and several appendixes. The main section contains three different levels of information to program the primary OCL cycles: LOAD, BUILD, and CALL.

Here is how to use each level of information:

- Use the **KEYWORD SEQUENCES** for an overall understanding of the OCL cycle.
- Use the **KEYWORD-RESPONSE SUMMARIES** for a quick recall of all the possible entries for each OCL statement.
- Use the **KEYWORD DESCRIPTIONS** when you need a detailed explanation of a particular keyword.

Keyword Sequences	shows the order of the OCL keywords for a cycle and indicates which keywords require responses.
Keyword-Response Summaries	lists keywords and possible responses for the three OCL cycles. In the Responses column: <ul style="list-style-type: none"> <li>● Words or letters in all capital letters (FORMS, BUILD, R1) represent actual entries.</li> <li>● Words or letters not in all capital letters (mmdyyy, Disk Name) represent information you must supply.</li> </ul>
Keyword Descriptions	gives detailed information about each keyword.

The appendixes contain information on programming OCL for complex jobs and special features or devices.

- A. Use *Appendix A: Advanced Topics for OCL* for information on:
  - Multi-volume files.
  - Including sort source or utility control statements in a procedure.
  - Increasing file size of the RPG procedure.
  - Entering RPG II source statements from the keyboard.
  - Chained procedures. The BUILD cycle is explained using the three levels of information used in the main OCL section.
- B. Use *Appendix B: Records-Tracks Conversion* for information on how to convert number of records to number of tracks.
- C. Use *Appendix C: Disk Organization* for information on how the disk packs are organized.
- D. Use *Appendix D: OCL for the 22" Printer* for information on using the optional 22 inch printer.
- E. Use *Appendix E: OCL for the IBM 2265-2 Display* for information on using the 2265-2 Display unit for OCL statements.
- G. Use *Appendix G: Card OCL for Model 6* for information using the online data recorder to enter OCL statements on cards.
- H. Use *Appendix H: OCL Error Messages* for detailed explanation of error messages printed during OCL prompting.
- I. Use *Appendix I: Co-Resident Systems* for information on using System/3 BASIC.





---

## SUMMARY OF CONVERSATIONAL OCL

---

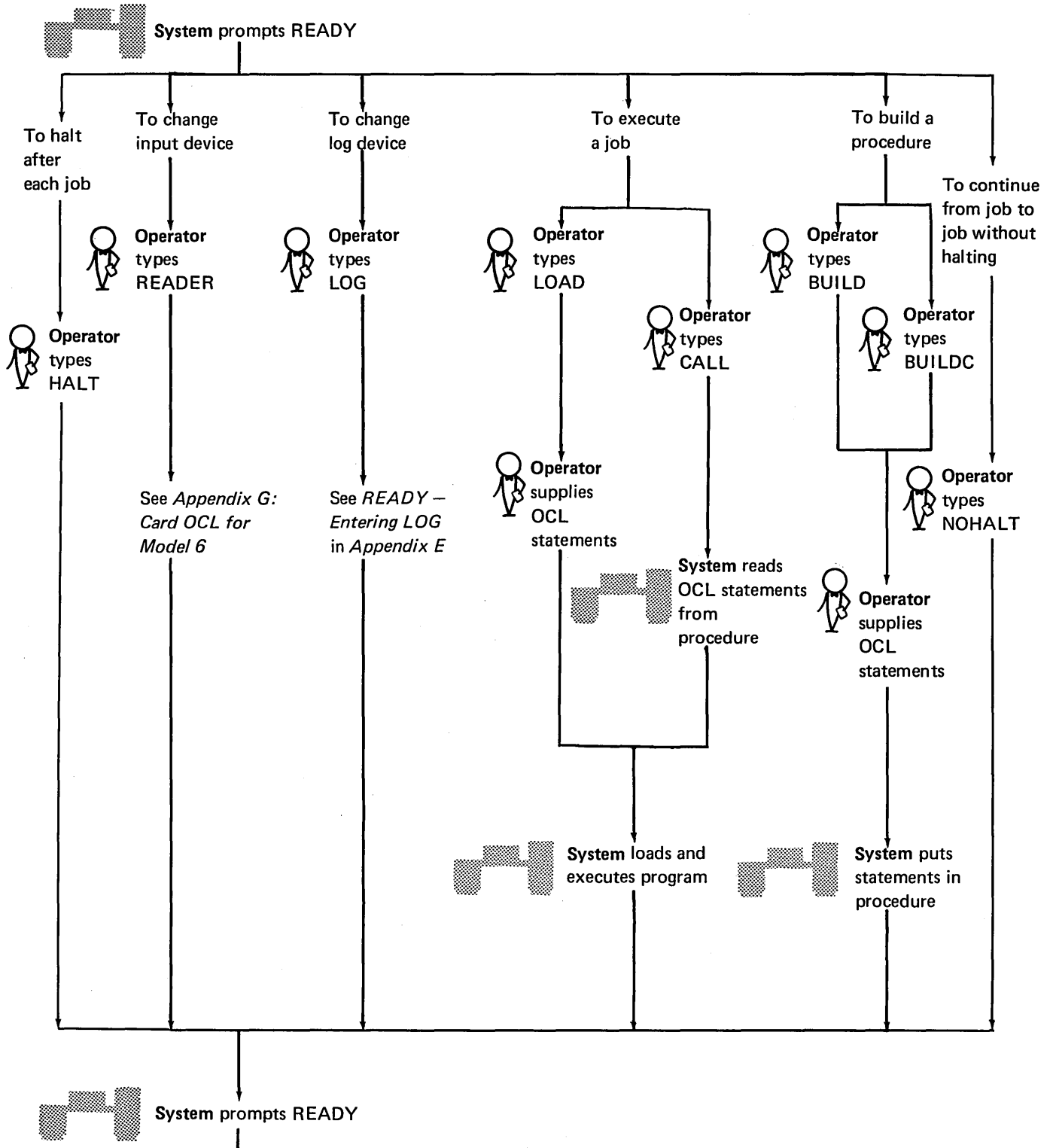
Every job run on the Model 6 requires a set of Operation Control Language (OCL) statements to give the system information about the job to be run (such as what program to use, what files to use, what job date to use, etc.). An OCL statement consists of a keyword and a response.

The OCL for Model 6 is called conversational OCL because it is really a conversation between the system and the operator. The system prints a keyword and waits for the operator to respond.

### THE JOB CYCLE

The system will prompt READY when it is ready to run jobs. (For information on preparing the system to run jobs, see the *IBM System/3 Model 6 Operator's Guide, GC21-7501*.) The response to READY tells the system what type of cycle you want to run.

# Job Cycle



## THE FOUR OCL CYCLES

There are four OCL cycles: LOAD, BUILD, CALL, and BUILDDC. The cycle you use depends on the type of job you're going to run.

Type of Job	OCL Cycle	Result
For jobs you want to run only a few times	LOAD	Provides the OCL statements necessary to run the job
For jobs you will run frequently	BUILD	Puts the OCL statements for a job into a source library procedure*
	CALL	Calls the procedure* from the source library
	BUILDDC**	Chains procedures*
<p>* A set of OCL statements in a source library is called a procedure.</p> <p>** See <i>Chained Procedure</i> in Appendix A.</p>		

### BUILD and CALL Cycles

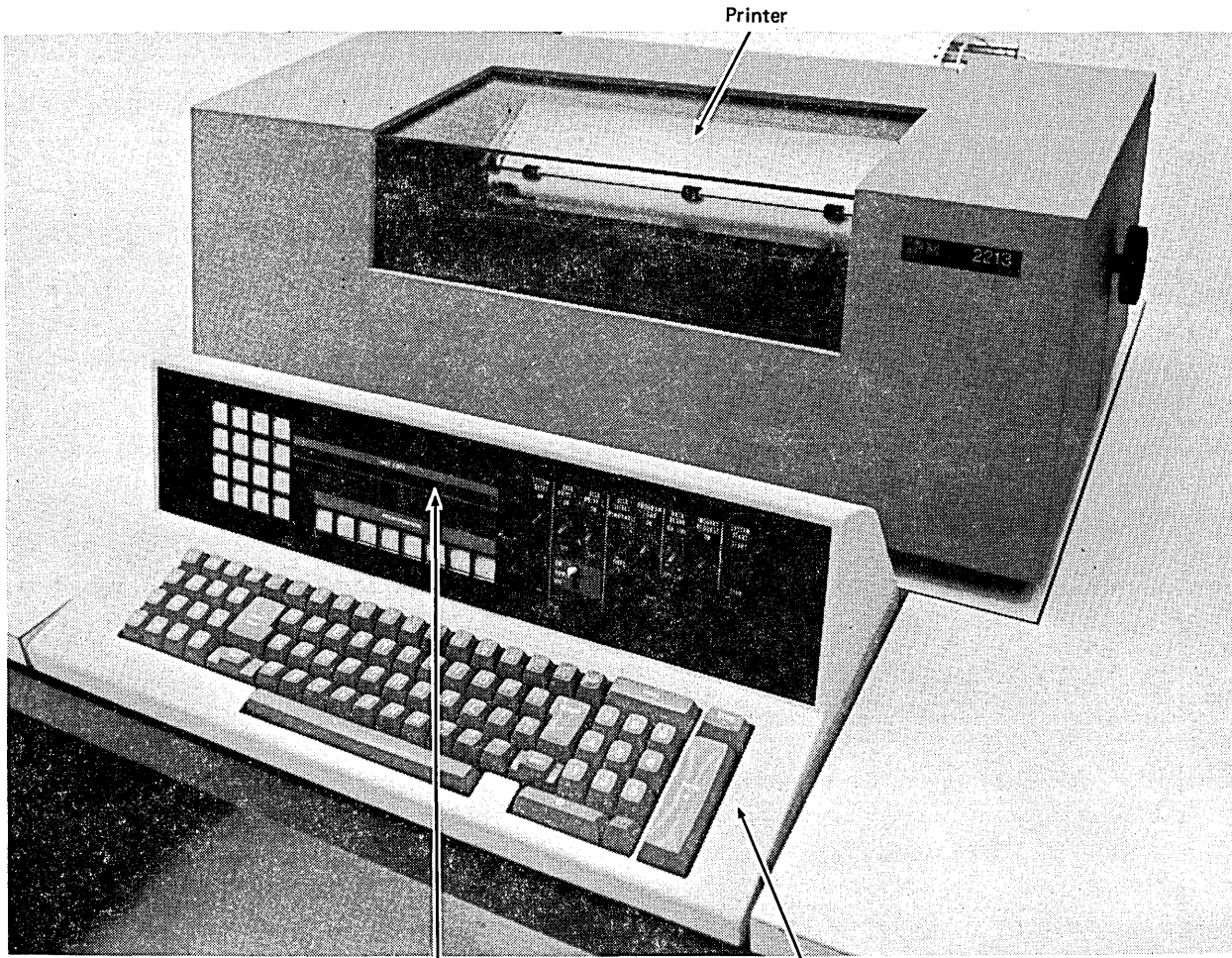
Using the BUILD and CALL cycles for jobs you run frequently saves operator time. Once the OCL statements for a job are put in a source library (with a BUILD cycle), you can request them (with a CALL cycle) anytime you want to run the job. Since the CALL cycle normally prompts only four keywords, the operator time involved is minimal.

#### Delayed Responses in the BUILD Cycle

Responding to a keyword by typing a question mark is referred to as a delayed response. Delayed responses are valid only in the BUILD cycle and only after keywords that contain a delayed response in the keyword-response chart (see *Keyword-Response Summary (Build Cycle)* in Appendix A). A delayed

response to any of these BUILD keywords will do the following:

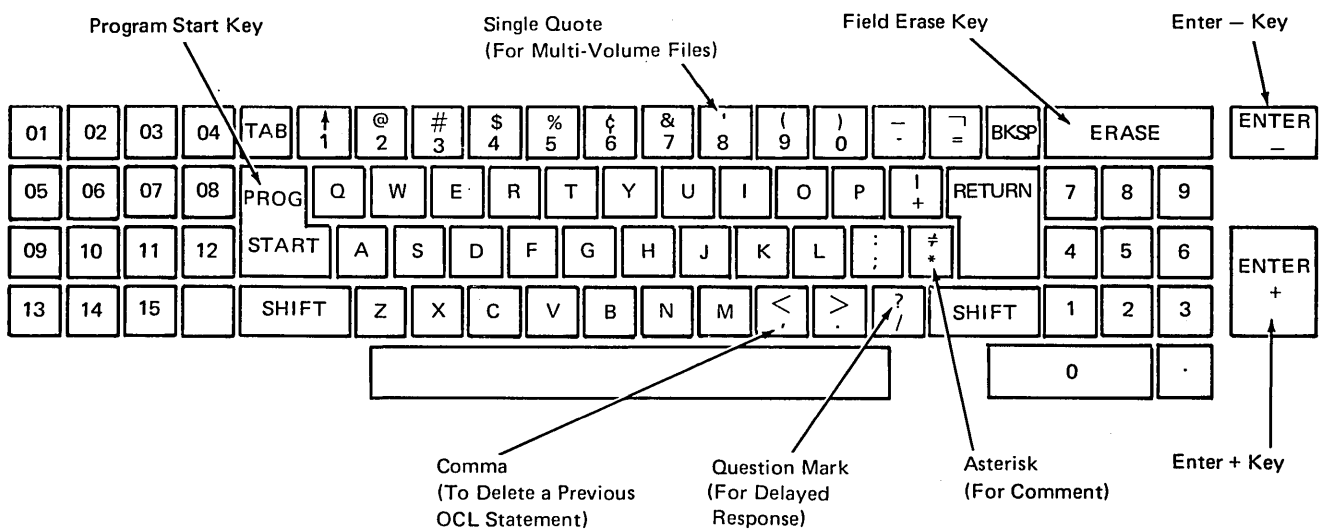
- Cause the system to reprompt the keyword during the CALL cycle.
- Force the operator to respond to the keyword when it is reprompted during the CALL cycle. (The system won't continue the CALL prompting cycle until the operator types a valid response.)



Halt Code Display

Keyboard

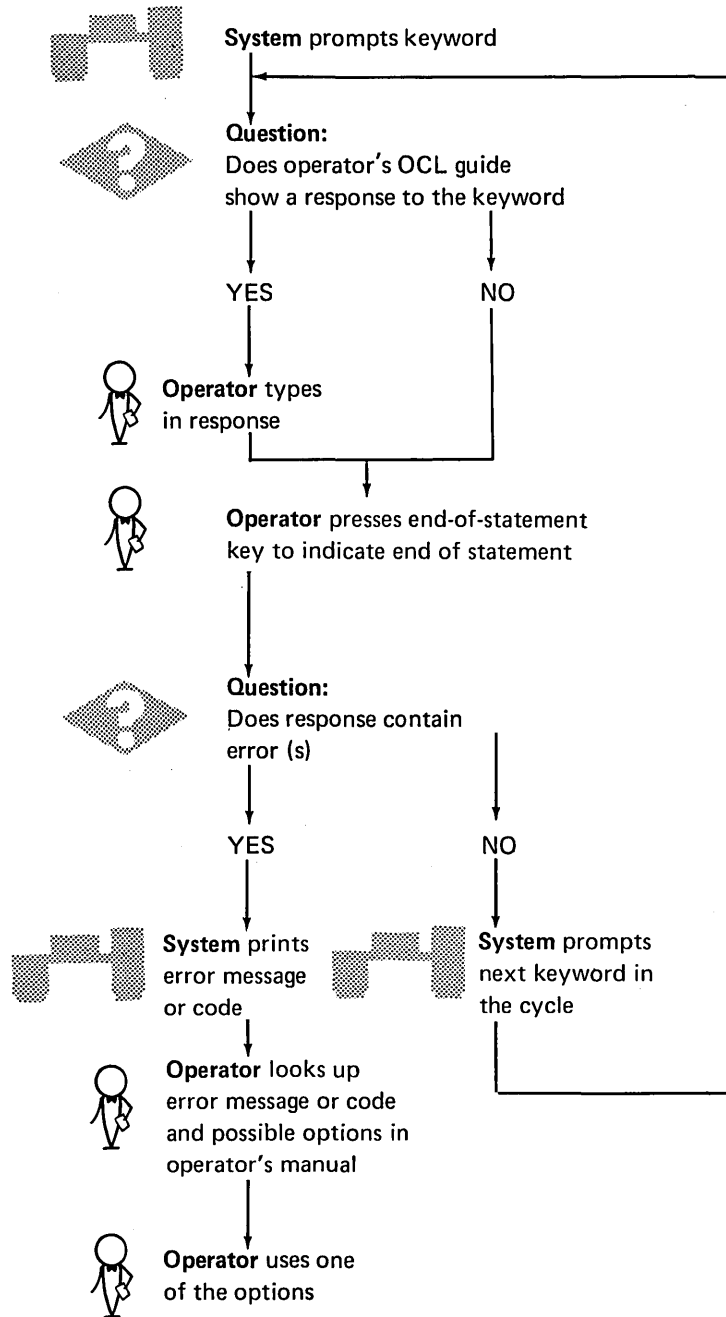
## Keyboard



## SYSTEM-OPERATOR INTERACTION DURING KEYWORD PROMPTING

The system analyzes the operator's response to each keyword. If the response contains a formatting error (such as invalid characters or duplicate procedure names), the system prints an error message and reprompts the keyword. Appendix H lists the error messages and a description of what caused the error. If the operator does not know the correct response, entering /\* as a response to any prompt will cancel the job and cause READY to be prompted.

# System-Operator Interaction During Keyword Prompting



## END-OF-STATEMENT KEYS

The operator must respond to each keyword that the system prompts. The operator responds to a keyword by typing the required information (if the keyword applies to the job) and by pressing an end-of-statement key. The end-of-statement key can be either PROG START or ENTER —. The Keyword-Response Summary charts in Appendix A explain the effect of end-of-statement keys on the prompting sequence.

### Program Start (PROG START) or Enter Plus (ENTER+)

Pressing the PROG START or ENTER+ key tells the system that the response is complete and to prompt the next keyword.

### Enter Minus (ENTER—)

Pressing the ENTER— key to end a response causes different processing depending on what keyword was prompted and what type of OCL cycle is being run.

*Pressing ENTER— After LOAD NAME or UNIT in a LOAD Cycle:* If the ENTER— key is used as an end-of-response to the LOAD NAME or UNIT prompts in a LOAD cycle, the remaining keywords in the cycle will be bypassed and MODIFY prompted.

*Pressing ENTER— After LOAD NAME or UNIT in a BUILD Cycle:* If the ENTER— key is used as an end-of-response to the LOAD NAME or UNIT prompts in a BUILD cycle, the system will prompt COMPILE OBJECT, SOURCE, or UNIT.

*Pressing ENTER— After FILE NAME:* If the ENTER— key is used as an end-of response to the FILE NAME prompt, the system prompts KEY LENGTH and HIKEY for multi-volume indexed files (see *Multi-Volume Files* in Appendix A).

*Pressing ENTER— in the File Keywords:* If the operator responds to FILE NAME, he must also respond to the next two file keywords: UNIT and PACK. He can, however, bypass any or all of the rest of the file keywords. To bypass a single keyword he presses the PROG START key as a response. To bypass all of the remaining file keywords he presses the ENTER— key either as an end-of-response or as a response. Pressing the ENTER— key causes the system to prompt FILE NAME for the next file.

## STATEMENT NUMBERS IN AN OCL CYCLE

Statement numbers are assigned by the system to statements in an OCL cycle. These statement numbers are used by the operator when using MODIFY to reference previous OCL statements.

Each OCL statement, except READY and MODIFY, is assigned a three digit number. The first number in a BUILD or CALL cycle is 000, and in a LOAD cycle 010.

The statement number is incremented by 10 for each major keyword (DATE, SWITCH, COMPILE OBJECT, FILE NAME, etc.), and by one for each minor keyword (UNIT, PACK, LABEL, RECORDS, etc.).

When the INCLUDE keyword is used to add utility control statements or sort source statements to a procedure, these included statements are assigned two-digit statement numbers. These statement numbers start with 00 and are incremented by one for each included statement.

The sample OCL jobs show the statement numbers assigned under various OCL cycles.

## COMMENTS

Comments can be entered after any response on the same line if at least one space is left between the response and the comment (see *Modify: Entering Comments* under MODIFY in Part I to add comments during MODIFY time).

## INQUIRY INTERRUPT

Certain programs can be interrupted while they are being processed. A request for interruption is called an inquiry request (made by depression of the inquiry key on the keyboard). Programs are usually interrupted to permit another program to run. Control is then given back to the first program.

The instructions given the compiler at compile time determine the inquiry type of a program.

The three types of programs include:

1. A program that cannot be interrupted (does not recognize an inquiry request).
2. A program that can be interrupted (does recognize an inquiry request). This is a B-type inquiry program.
3. An inquiry program that can only be executed when an inquiry request is made. This is an I-type program.

Usually I-type programs are read in only when a program is interrupted. In this case the inquiry program will not recognize an inquiry request. However, if an inquiry program is loaded in the normal manner (not because of a program interrupt), it can only be executed when an inquiry request is made. While this program is running, it will not recognize an inquiry request.

The inquiry interrupt involves these three steps:

1. When the program recognizes an inquiry request, a Roll-Out routine moves the interrupted program from main storage to disk.
2. The program for which the interrupt was requested must be loaded normally. The interrupting program may be any type. This interrupting program cannot be interrupted.
3. After the interrupting program is executed, the interrupted program moves back into main storage using a Roll-In routine. The interrupted program begins execution at the point of interruption and terminates in a normal manner.

The *IBM System/3 Model 6 RPG II Reference Manual, SC21-7517*, describes coding necessary to define inquirable programs.

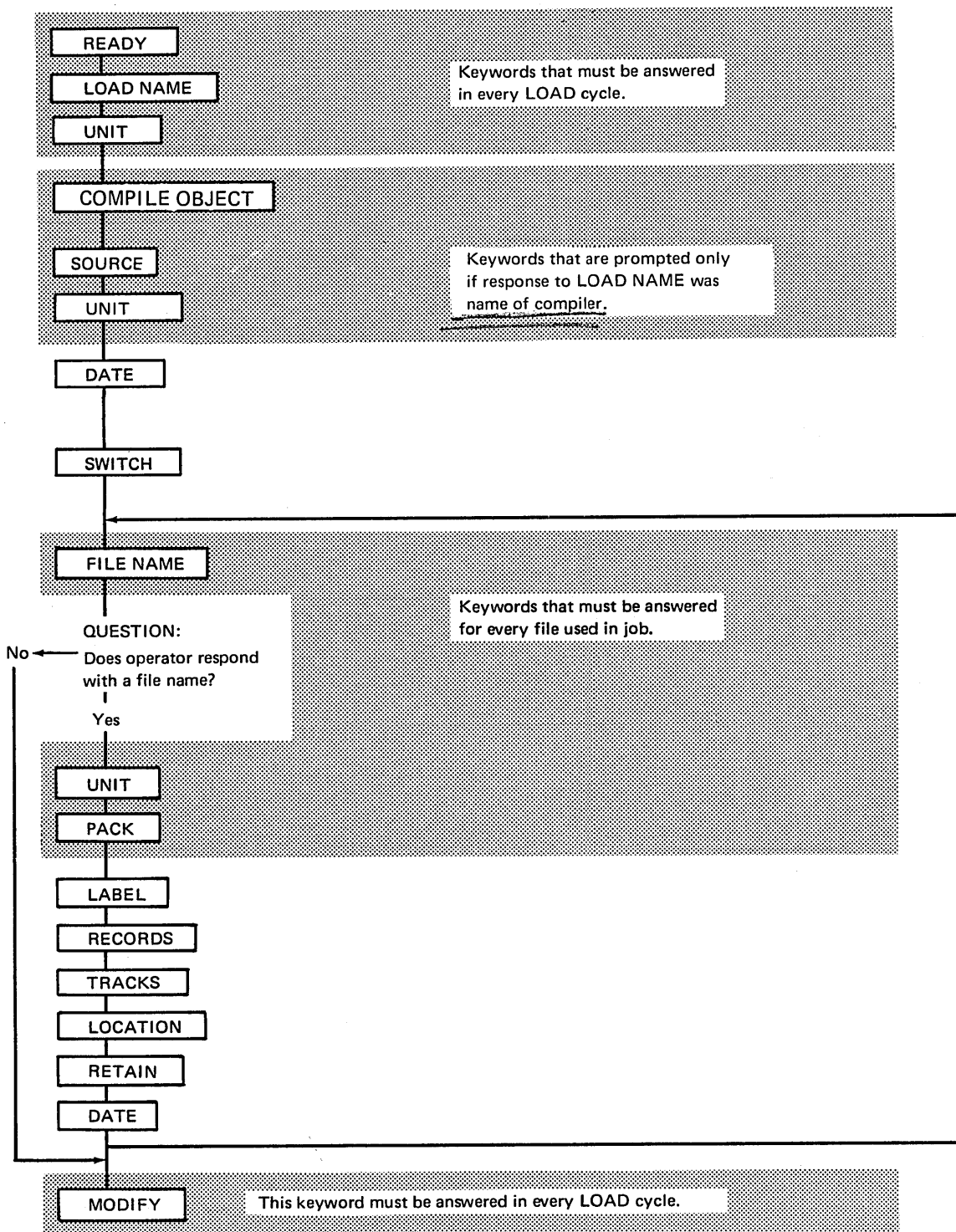
### Restrictions During Inquiry

Inquiry always causes the conversational OCL scheduler to be used, even if the interrupted program was running under the card scheduler. The OCL statements cannot be read from cards during inquiry.

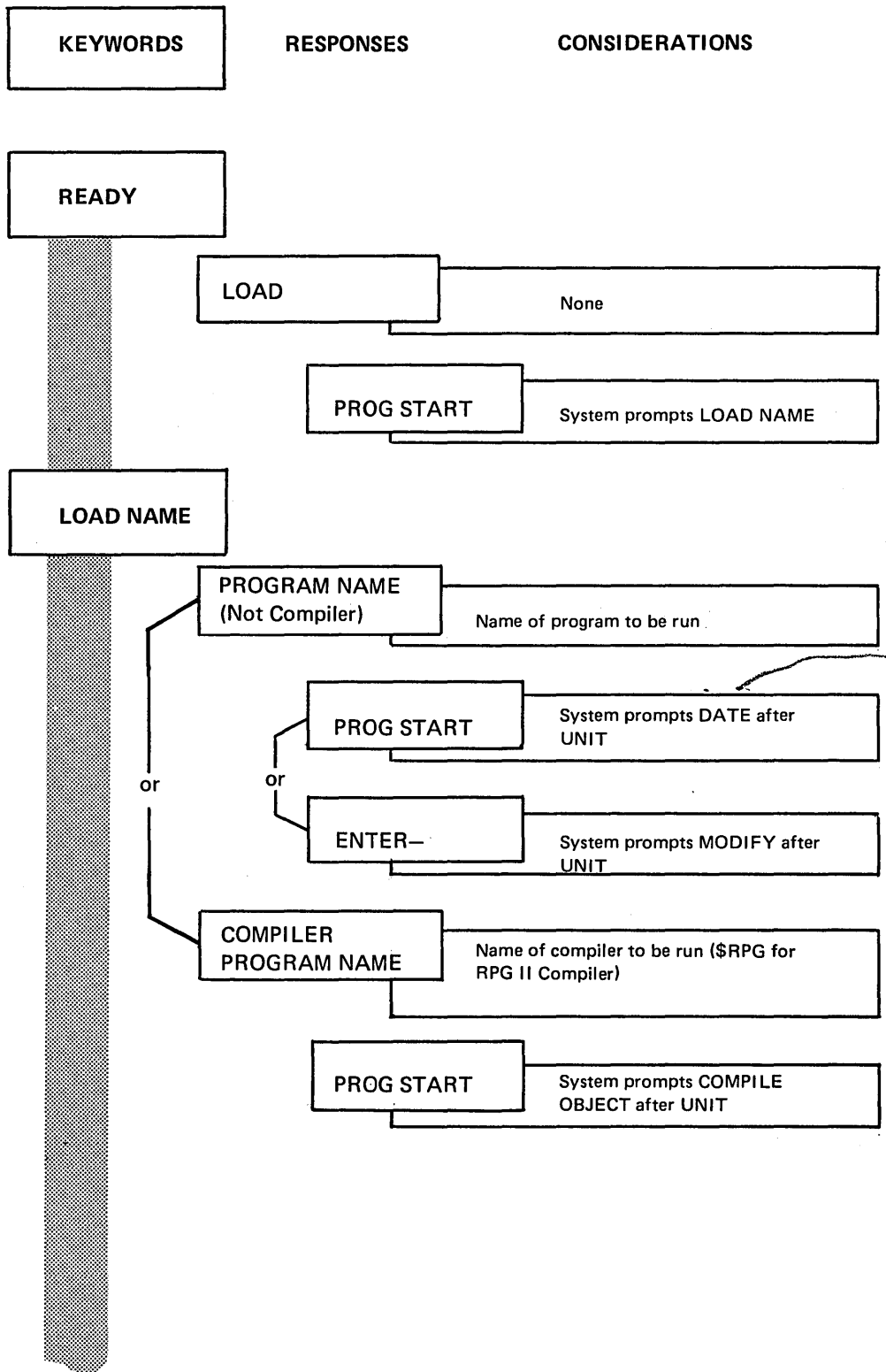
The Log device cannot be changed during inquiry.



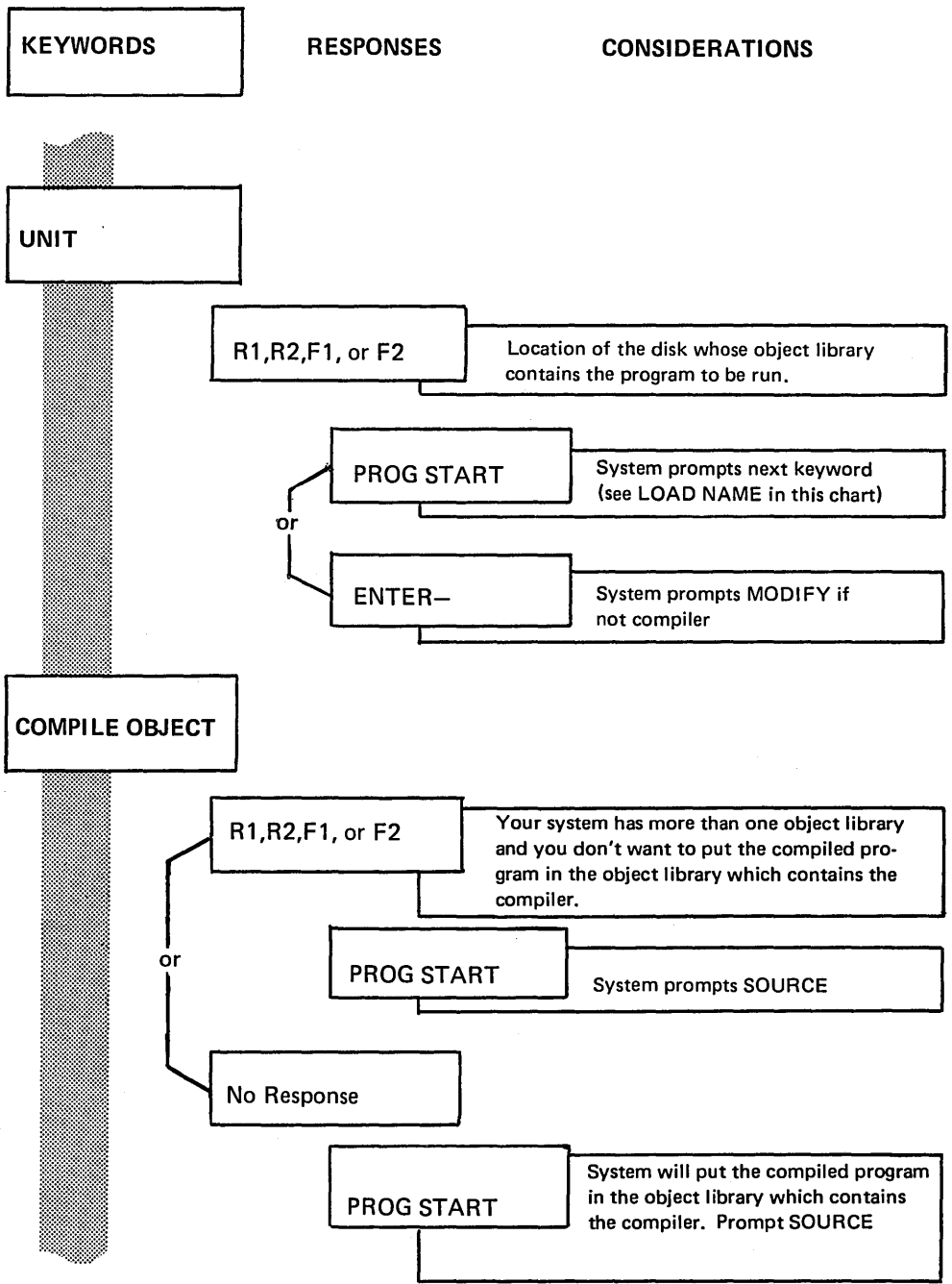
# Keyword Sequence for OCL Load Cycle



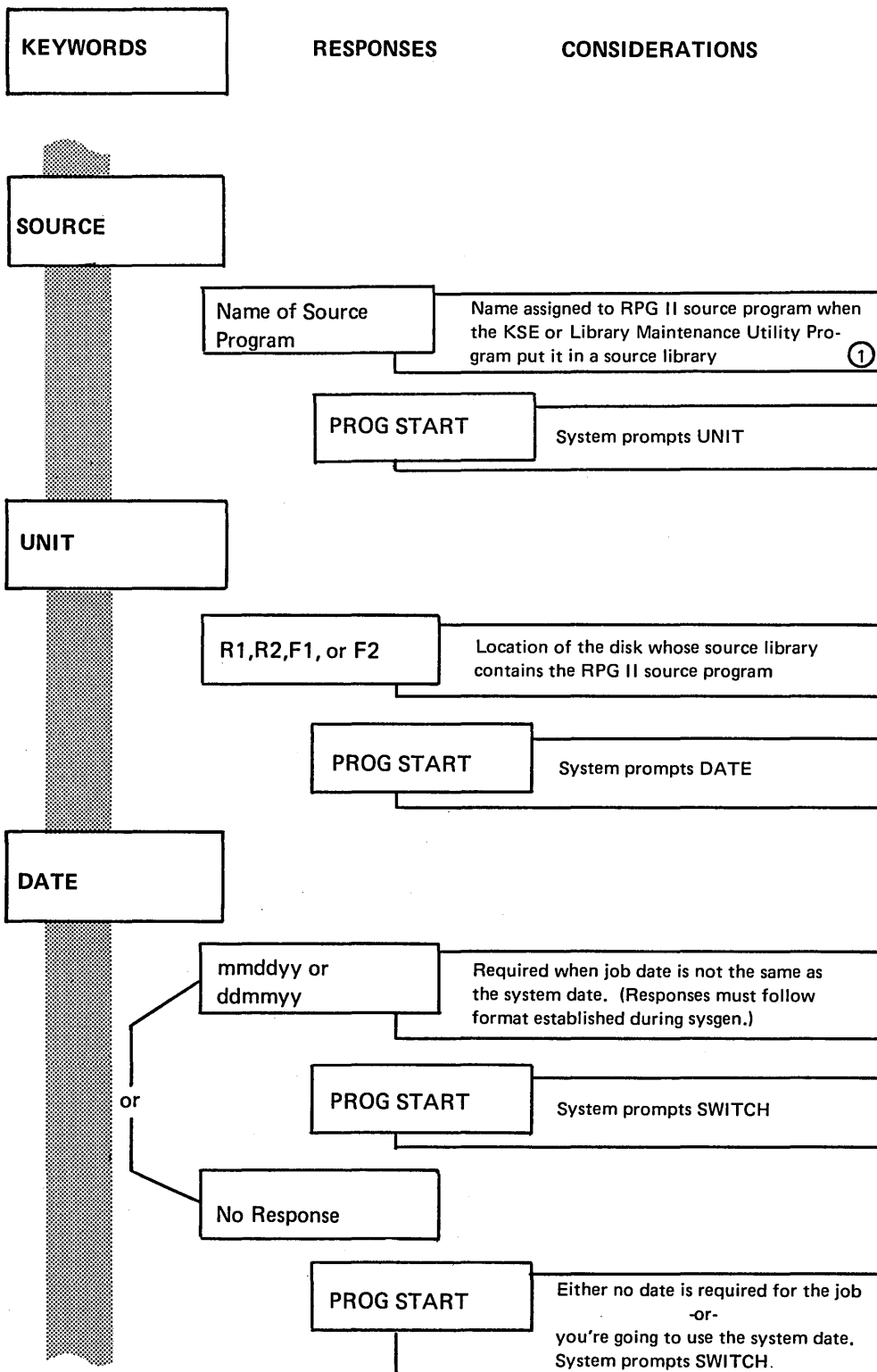
# Keyword-Response Summary (Load Cycle)



PG 17



*DATE NEXT*



① For information about the KSE Program see the *IBM System/3 Model 6 Conversational Utility Programs Reference Manual*, SC21-7528.

For information about the Library Maintenance Program see Part II of this manual.

**KEYWORDS**

**RESPONSES**

**CONSIDERATIONS**

**SWITCH  
(XXXXXXXX)**

8-position setting  
(combination of  
1's, 0's, and X's)

Required to change external indicators in RPG programs. Three choices for each position:  
1 = turn indicator on  
0 = turn indicator off  
X = leave indicator as is

or

**PROG START**

System prompts FILE NAME

No Response

**PROG START**

Job does not use external indicators or you want to use the current setting. System prompts FILE NAME

**FILE NAME**

File name of file  
used by program

Columns 7-14 of RPG File Description Specifications, or predefined file name for system programs

**PROG START**

System prompts UNIT

or

**ENTER-**

System prompts KEY LENGTH  
(see *Multi-Volume Files* in  
Appendix A)

or

No Response

**PROG START**

Either your job uses no files at all  
-or-  
you have already described all the  
files the job uses. You want the  
system to prompt MODIFY

**KEYWORDS**

**RESPONSES**

**CONSIDERATIONS**

**UNIT**

R1,R2,F1, or F2

During a file creation run –  
location of disk where you want  
to write the file.  
During other runs – location of disk which  
contains the file to be processed

PROG START

System prompts PACK

**PACK**

*G CHAN*  
Disk Name

During a file creation run – the name which  
identifies the disk on which you want to  
write the file.  
During other runs – name which identifies  
the disk on which the file is located

PROG START

System prompts LABEL

or

ENTER–

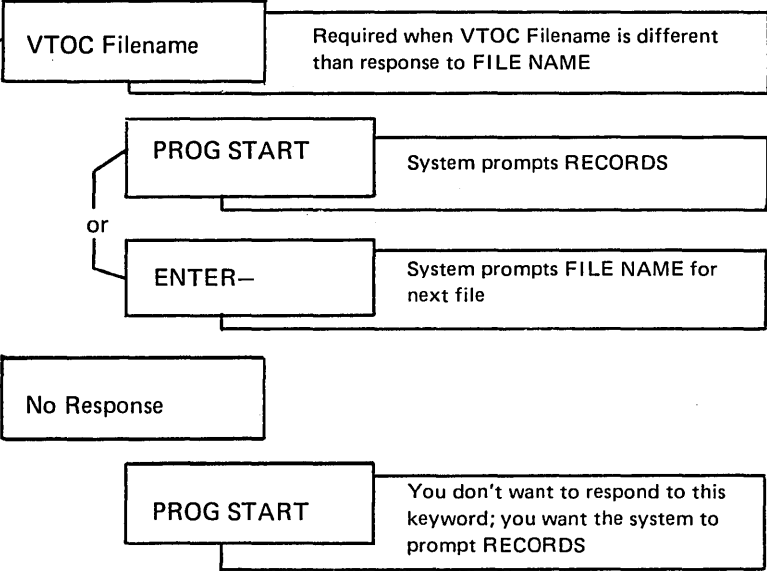
System prompts FILE NAME for  
next file

**KEYWORDS**

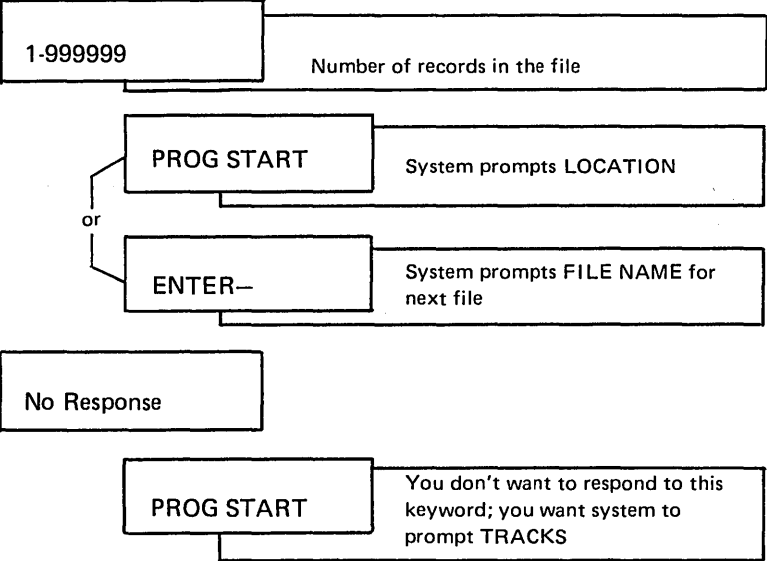
**RESPONSES**

**CONSIDERATIONS**

**LABEL**



**RECORDS ①**



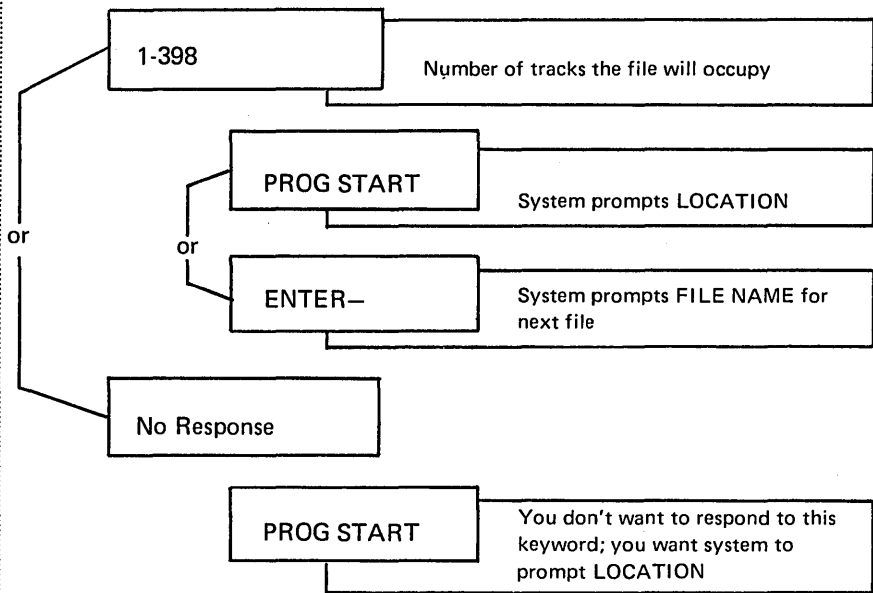
① At file creation time, *either* the number of records *or* the number of tracks must be specified.

**KEYWORDS**

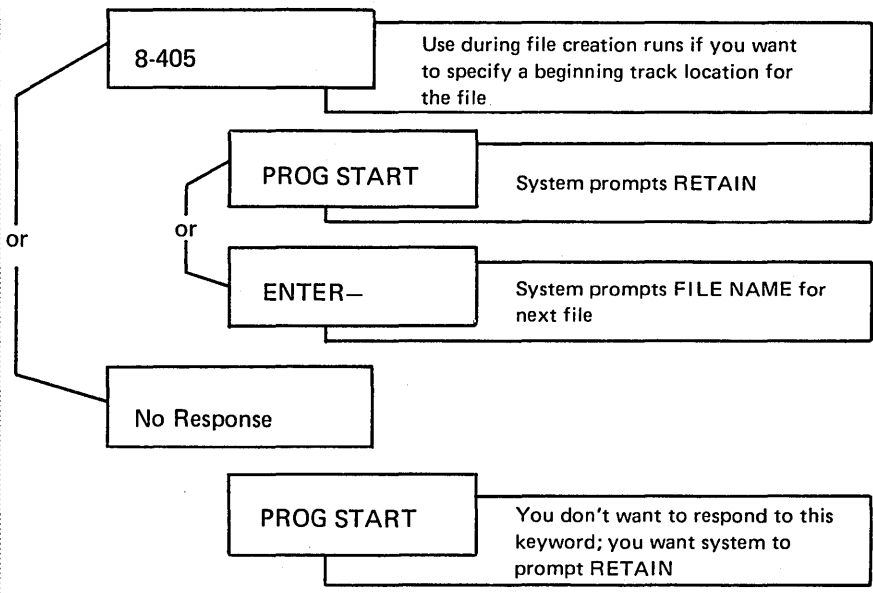
**RESPONSES**

**CONSIDERATIONS**

**TRACKS ①**



**LOCATION**



① At file creation time, *either* the number of records *or* the number of tracks must be specified. If operator entered number of RECORDS, TRACKS will not be prompted.

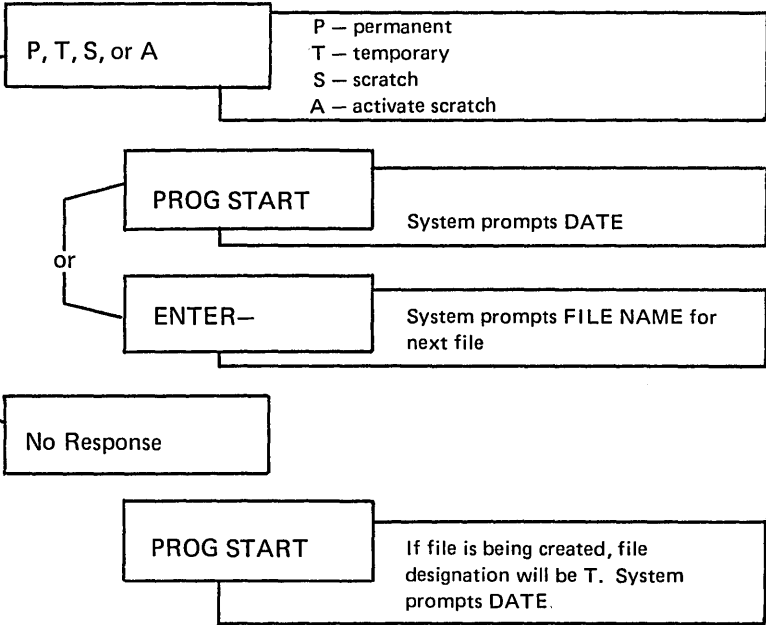


**KEYWORDS**

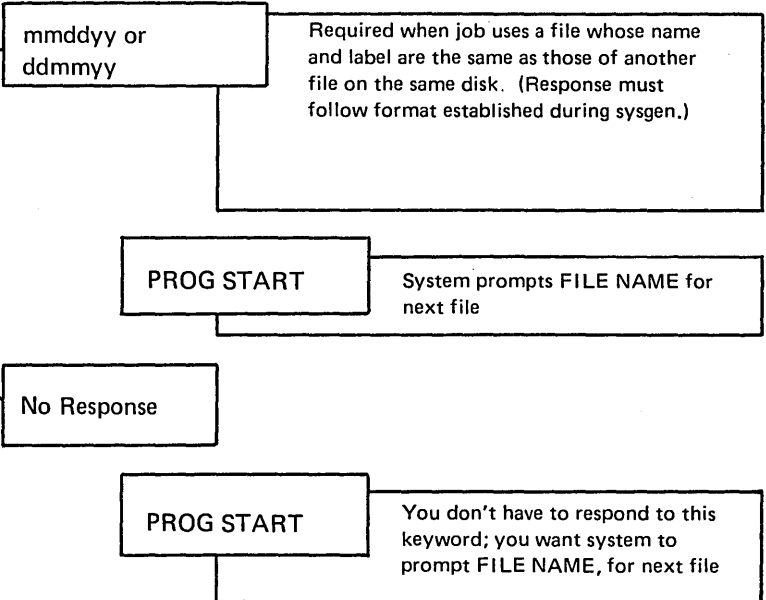
**RESPONSES**

**CONSIDERATIONS**

**RETAIN**



**DATE**



**KEYWORDS**

**RESPONSES**

**CONSIDERATIONS**

**MODIFY**  
(Operator can use one, all, or a combination of the responses.)

LOG

Used only if CRT display or 22" printer on system (see Appendixes D and E)

PROG START

System prompts LOG DEVICE

CANCEL

Cancel job

PROG START

System prompts READY or displays end-of-job halt

FORMS

Change lines per page printed output for system programs

PROG START

System prompts FORMS DEVICE

Asterisk (\*)  
Followed by comments

Enter comment.

PROG START

System waits for next MODIFY response

Statement number and comma

To delete statement

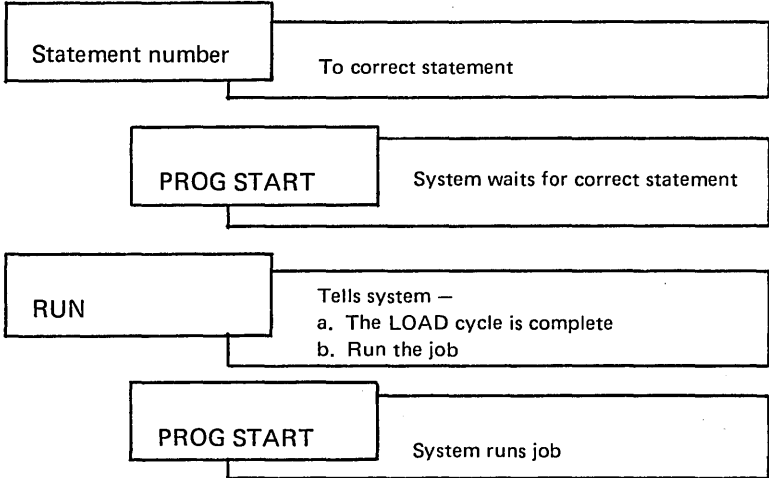
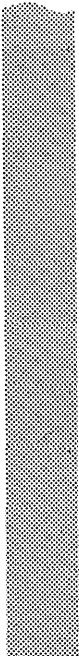
PROG START

System waits for next MODIFY response

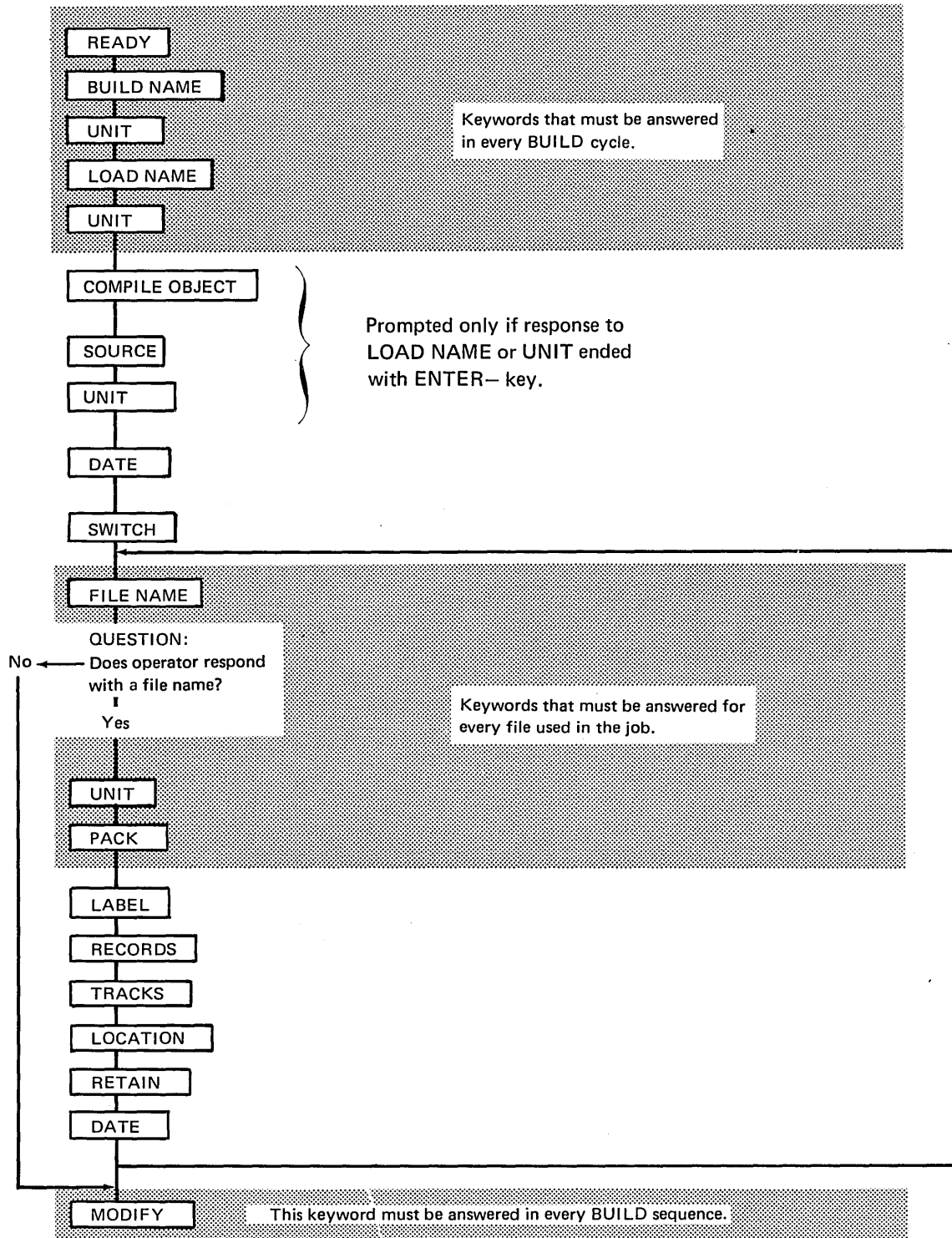
**KEYWORDS**

**RESPONSES**

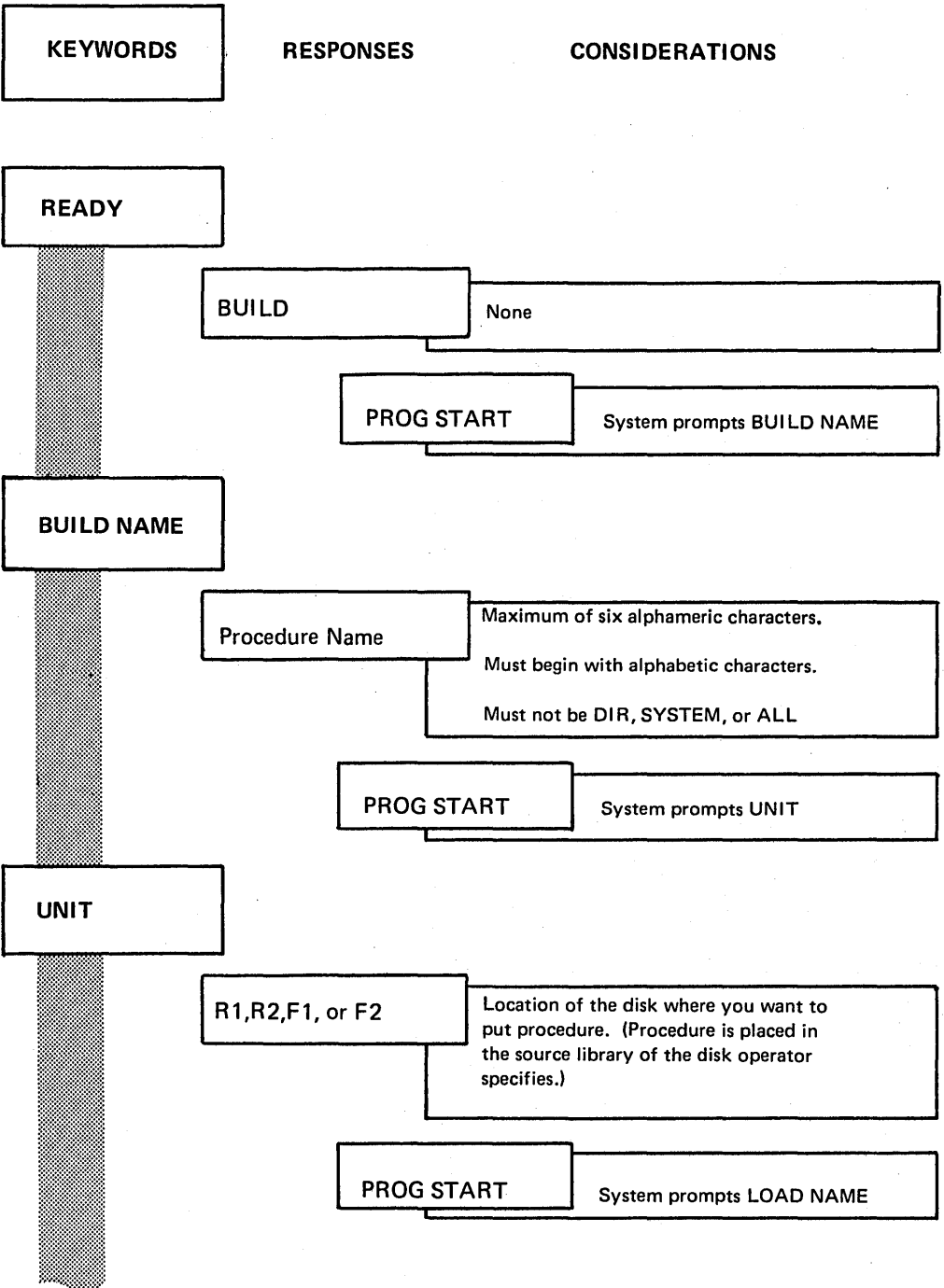
**CONSIDERATIONS**

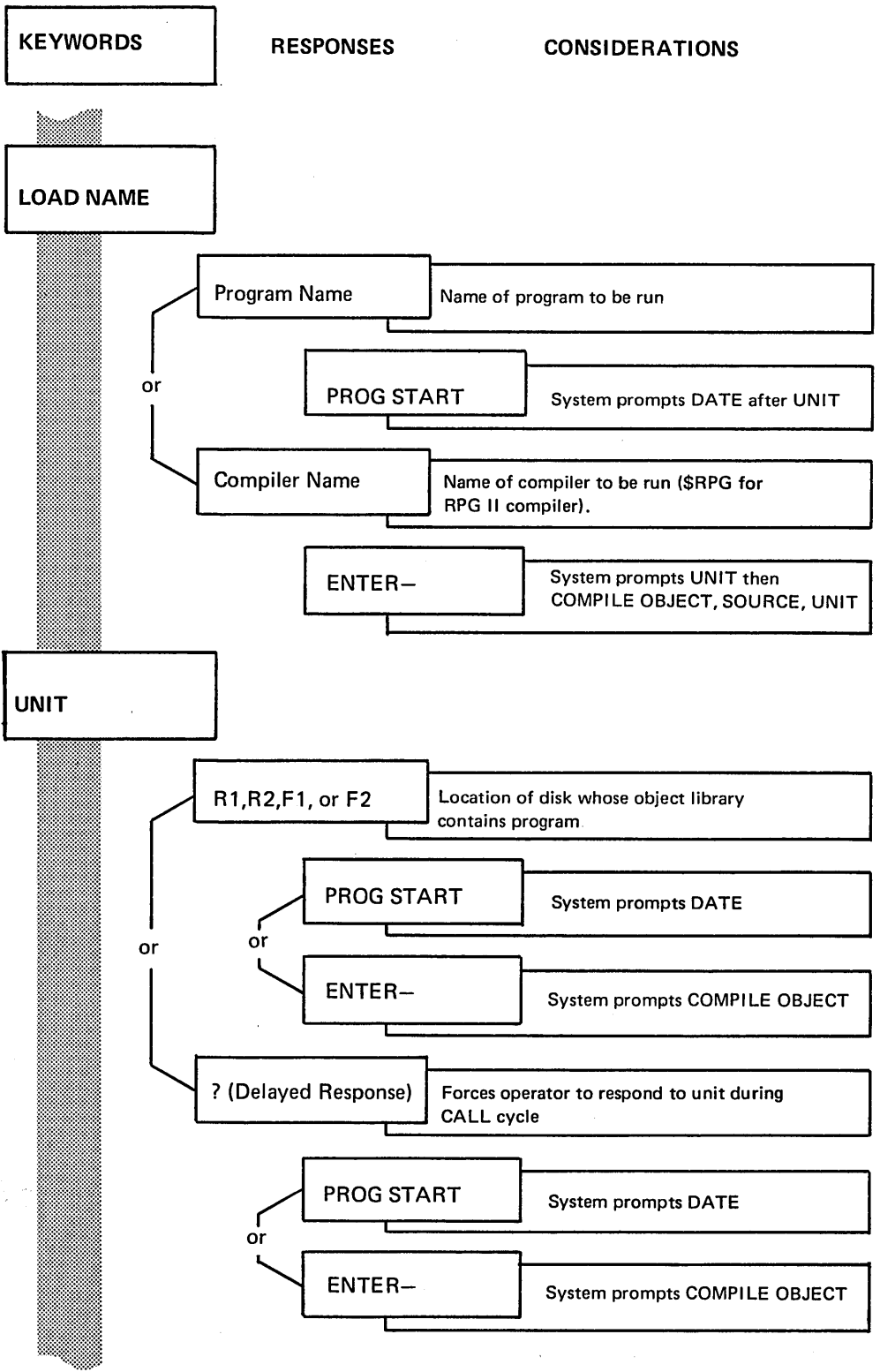


# Keyword Sequence for OCL Build Cycle



# Keyword-Response Summary (Build Cycle)



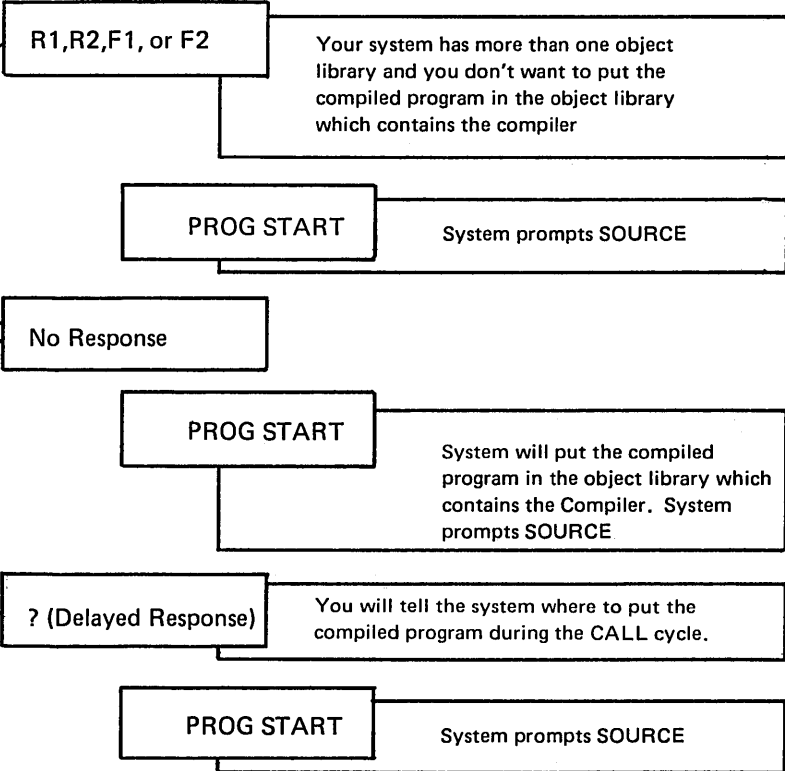


**KEYWORDS**

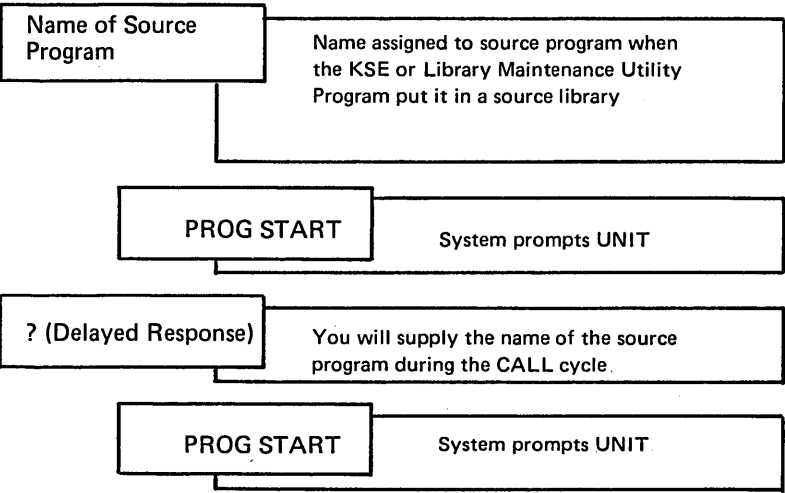
**RESPONSES**

**CONSIDERATIONS**

**COMPILE OBJECT**



**SOURCE**



*only if  
RPG 15  
COAD  
NAME*

**KEYWORDS**

**RESPONSES**

**CONSIDERATIONS**

**UNIT**

R1,R2,F1, or F2

Location of the disk whose source library contains the RPG source program

or

PROG START

System prompts DATE

? (Delayed Response)

You will supply the location of the source program during the CALL cycle

PROG START

System prompts DATE

**DATE**

mmdyy or ddmmy

To put a job date in the procedure (Response must follow format established during system.)

or

PROG START

System prompts SWITCH

? (Delayed Response)

Forces operator to supply DATE during CALL cycle

or

PROG START

System prompts SWITCH

No Response

PROG START

If no date is necessary for job or system date is acceptable. DATE will not be part of procedure

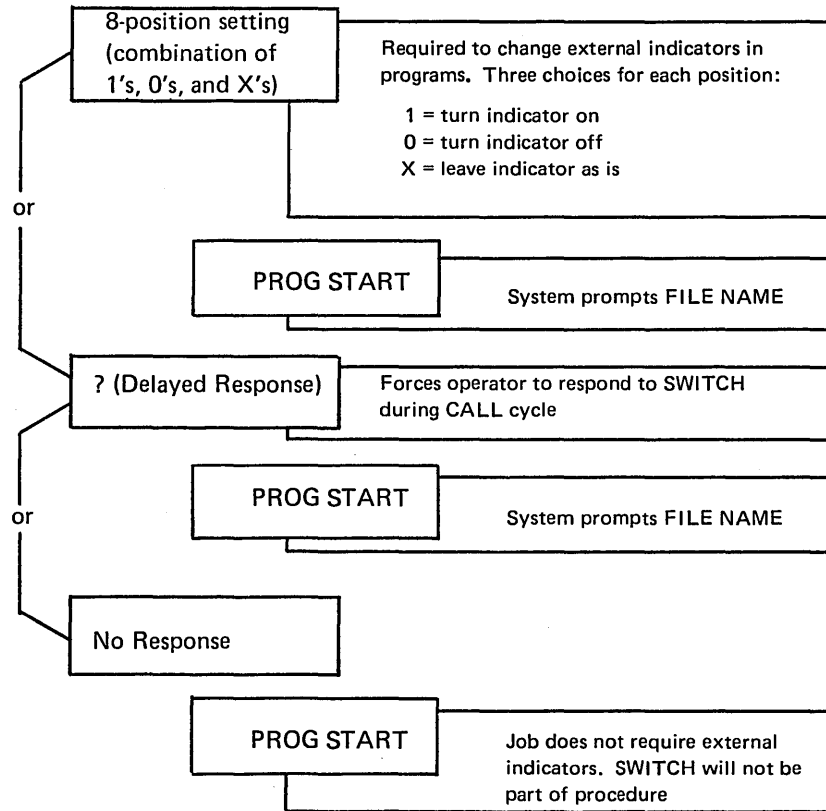


**KEYWORDS**

**RESPONSES**

**CONSIDERATIONS**

**SWITCH**

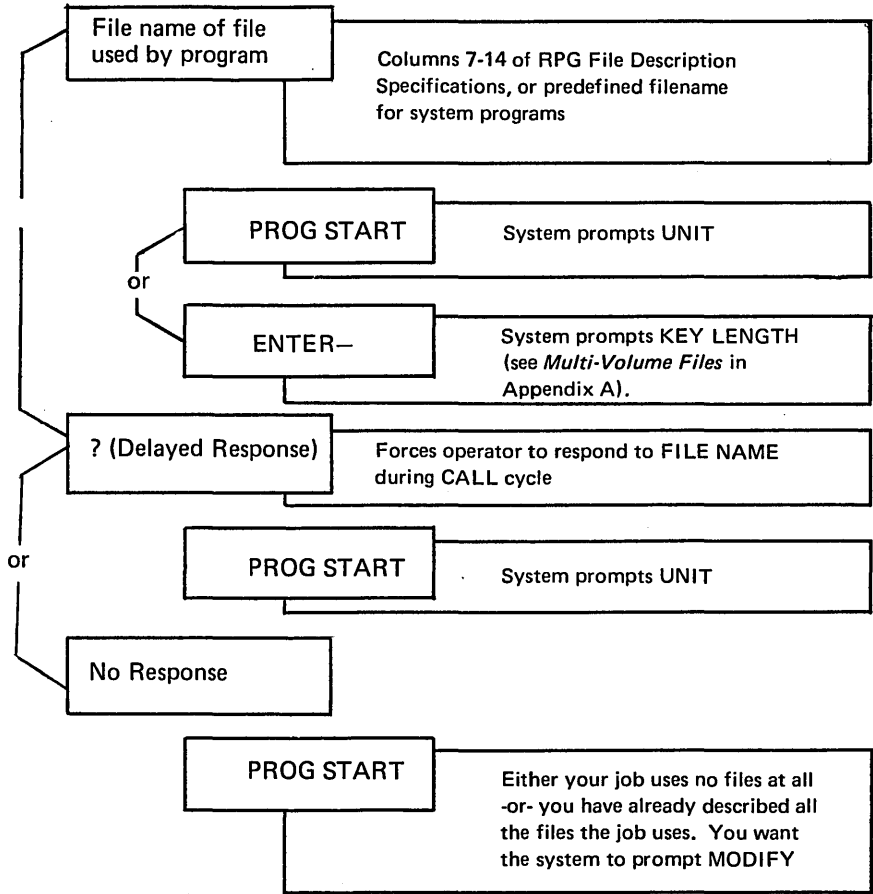


**KEYWORDS**

**RESPONSES**

**CONSIDERATIONS**

**FILE NAME**

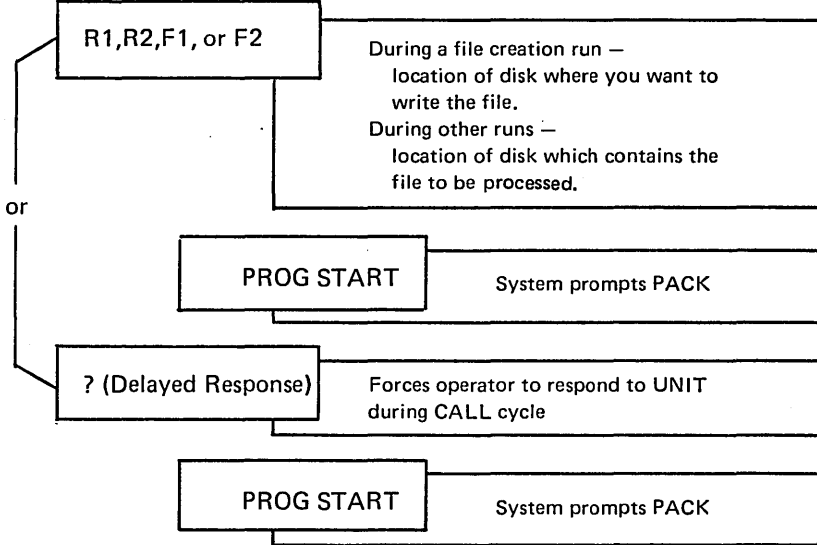


**KEYWORDS**

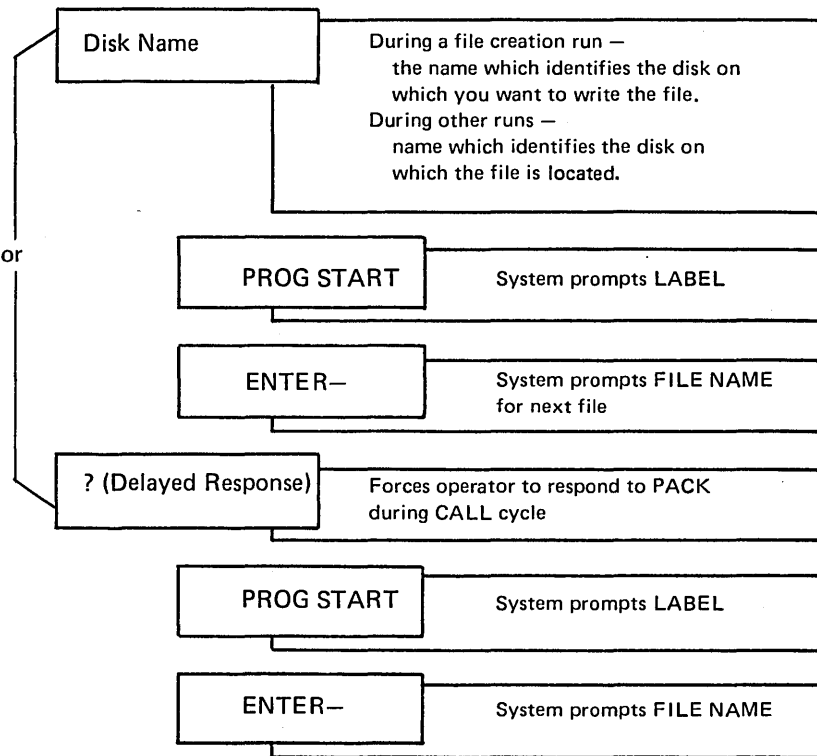
**RESPONSES**

**CONSIDERATIONS**

**UNIT**



**PACK**

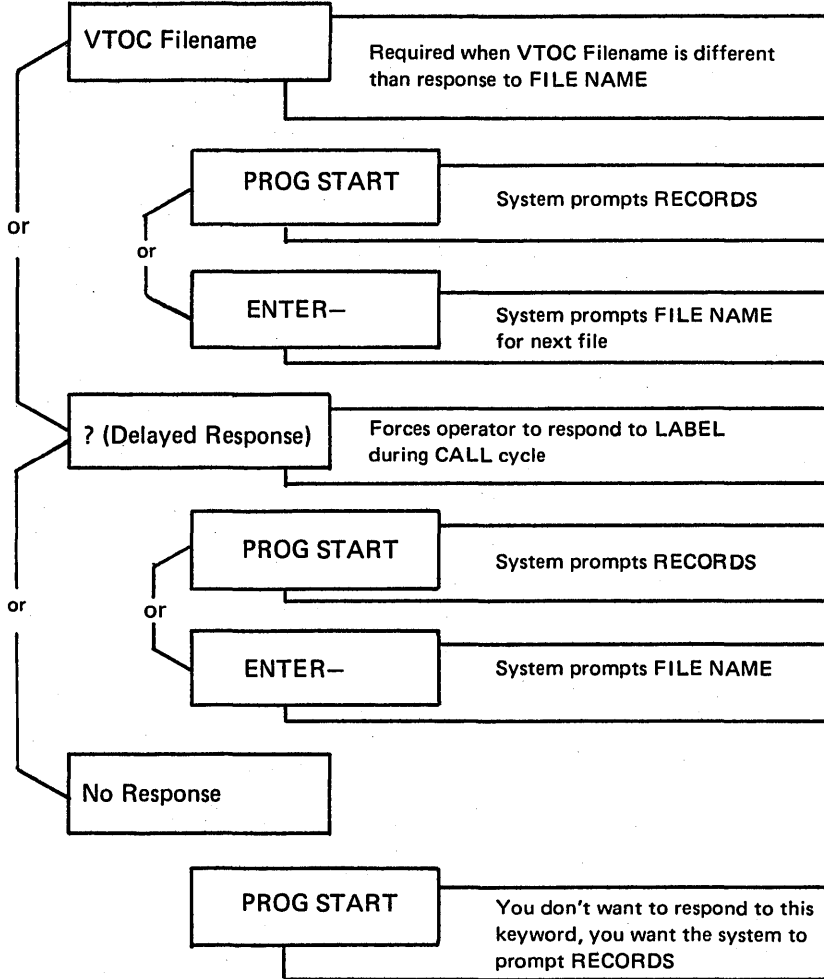


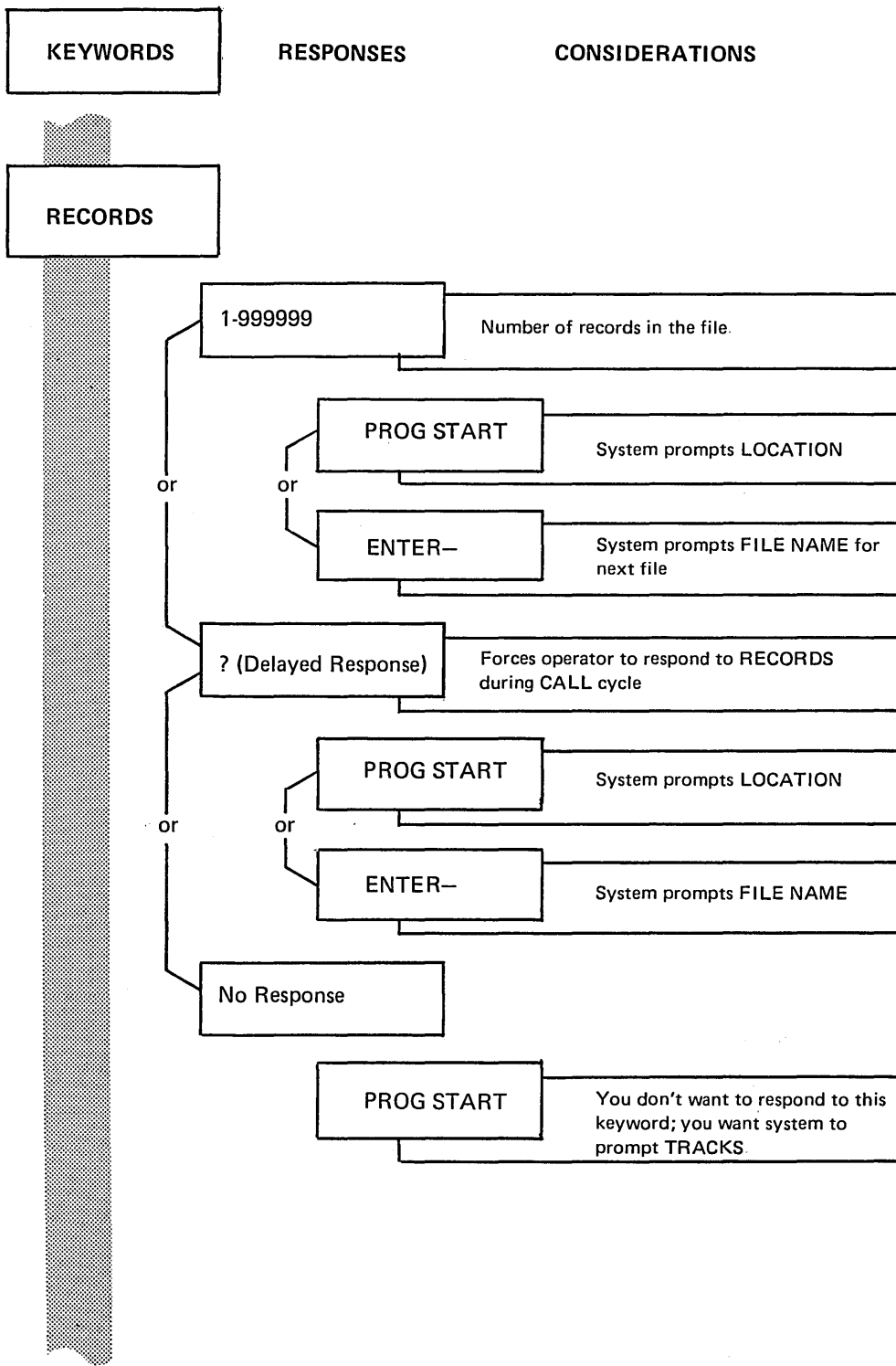
**KEYWORDS**

**RESPONSES**

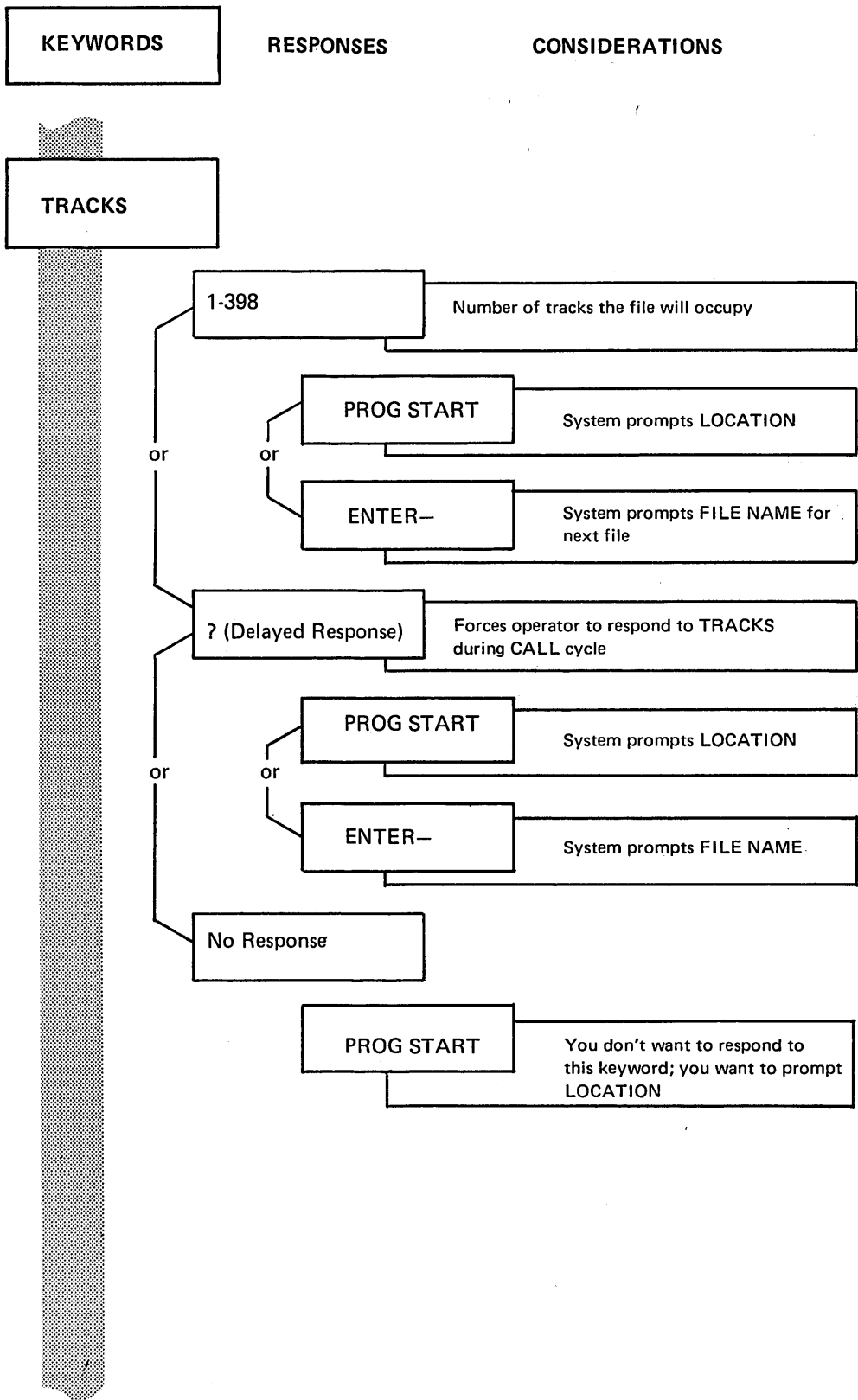
**CONSIDERATIONS**

**LABEL**





When a file is created, either the number of records *or* the number of tracks must be specified. If operator entered number of RECORDS, TRACKS will not be prompted.



When a file is created, either the number of records *or* the number of tracks must be specified. If operator entered number of RECORDS, TRACKS will not be prompted.

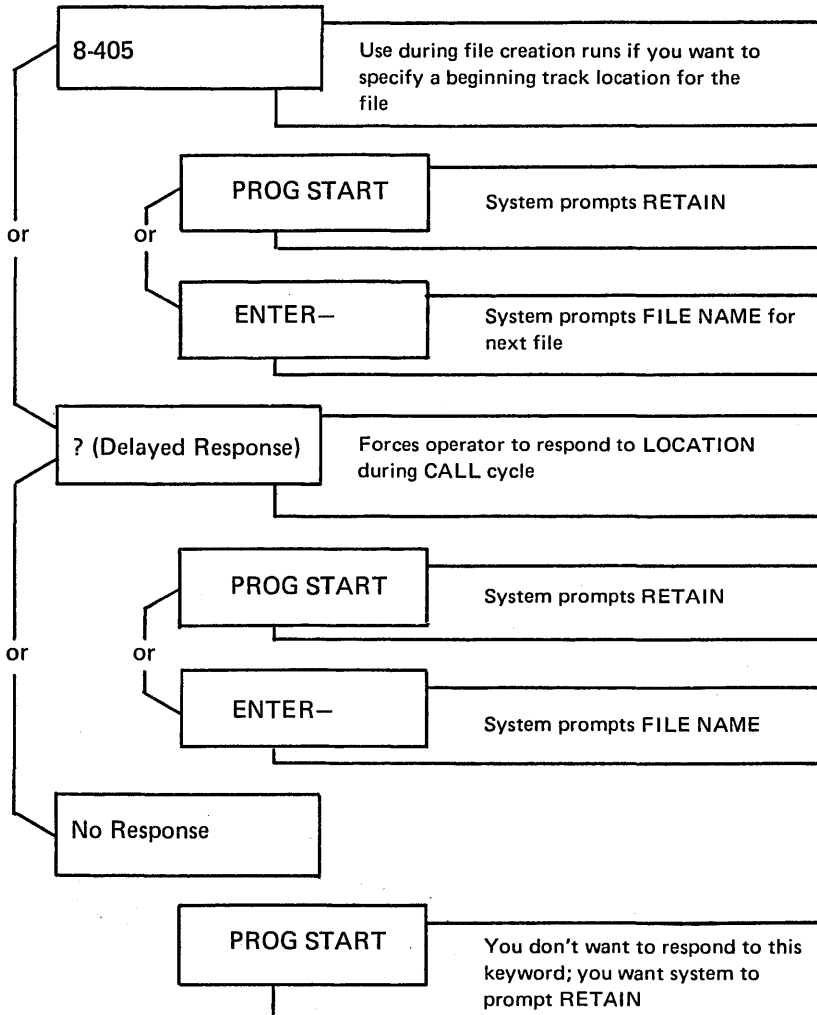
KEYWORDS

RESPONSES

CONSIDERATIONS

LOCATION

1. DISTINGUISH BETWEEN 2 FILES WITH SAME LABEL  
OR  
2. FILE CREATION



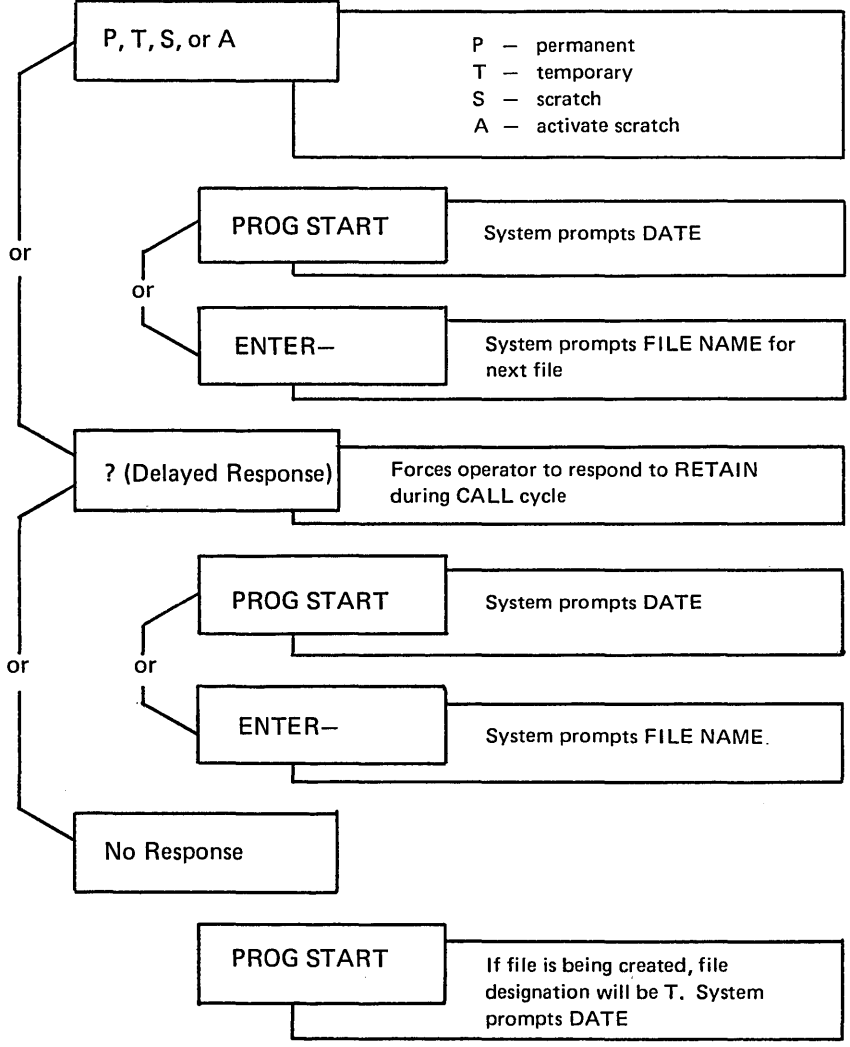
**KEYWORDS**

**RESPONSES**

**CONSIDERATIONS**

*- USE ONLY IN CREATING NEW FILE*

**RETAIN**





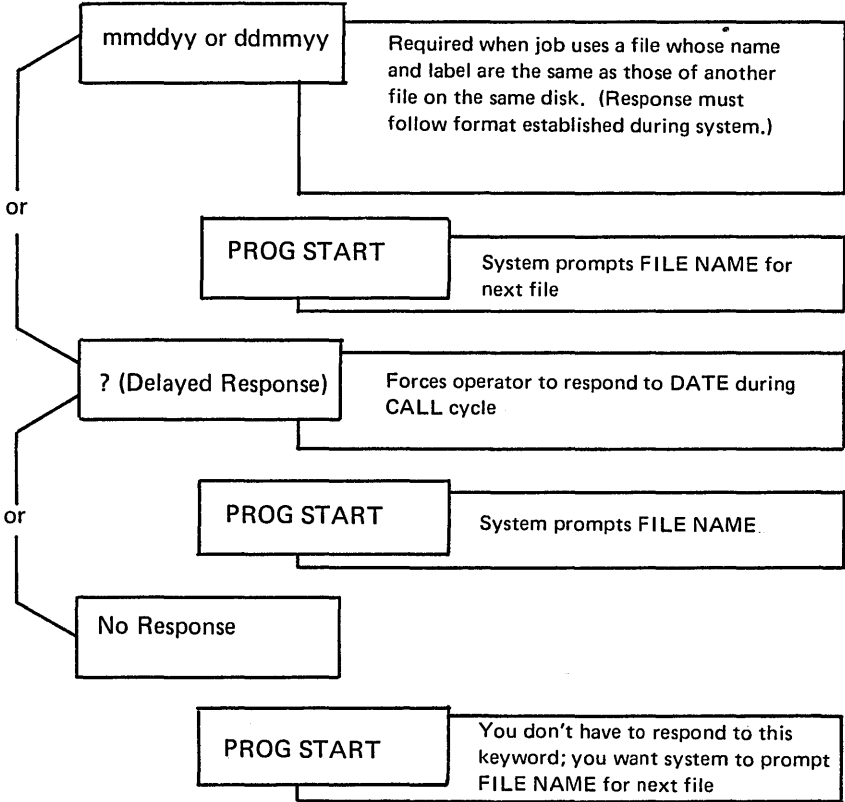
**KEYWORDS**

**RESPONSES**

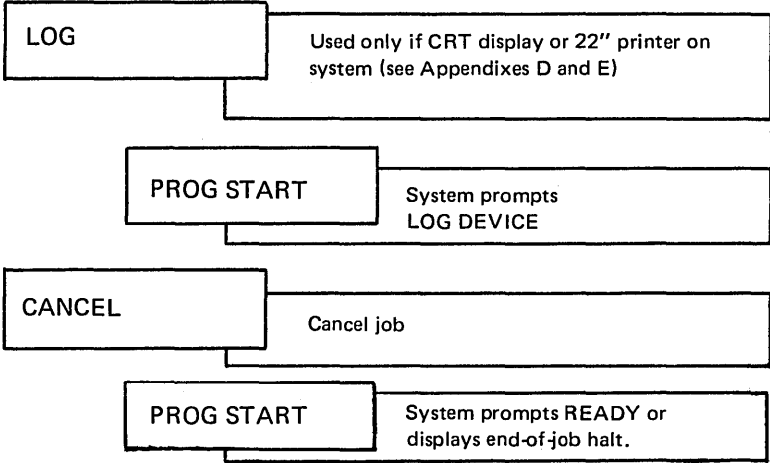
**CONSIDERATIONS**

**DATE**

*USE ONLY WHEN EXISTING FILE ACCESSING AND LOCATION WASN'T USED.*



**MODIFY**  
(Operator can use one, all, or a combination of the responses.)



**KEYWORDS****RESPONSES****CONSIDERATIONS****FORMS**Change lines per page printed  
output for system programs**PROG START**

System prompts FORMS DEVICE

**Asterisk (\*) Followed  
by Comments**

Enter comment

**PROG START**System waits for next MODIFY  
response**Statement number  
and comma**

To delete statement

**PROG START**System waits for next MODIFY  
response**Statement number**

To correct statement

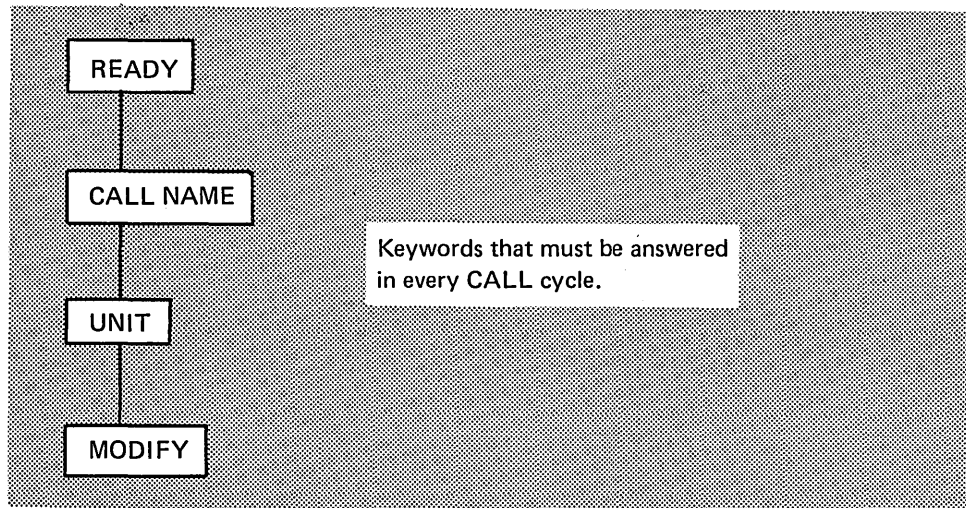
**PROG START**

System waits for correct statement

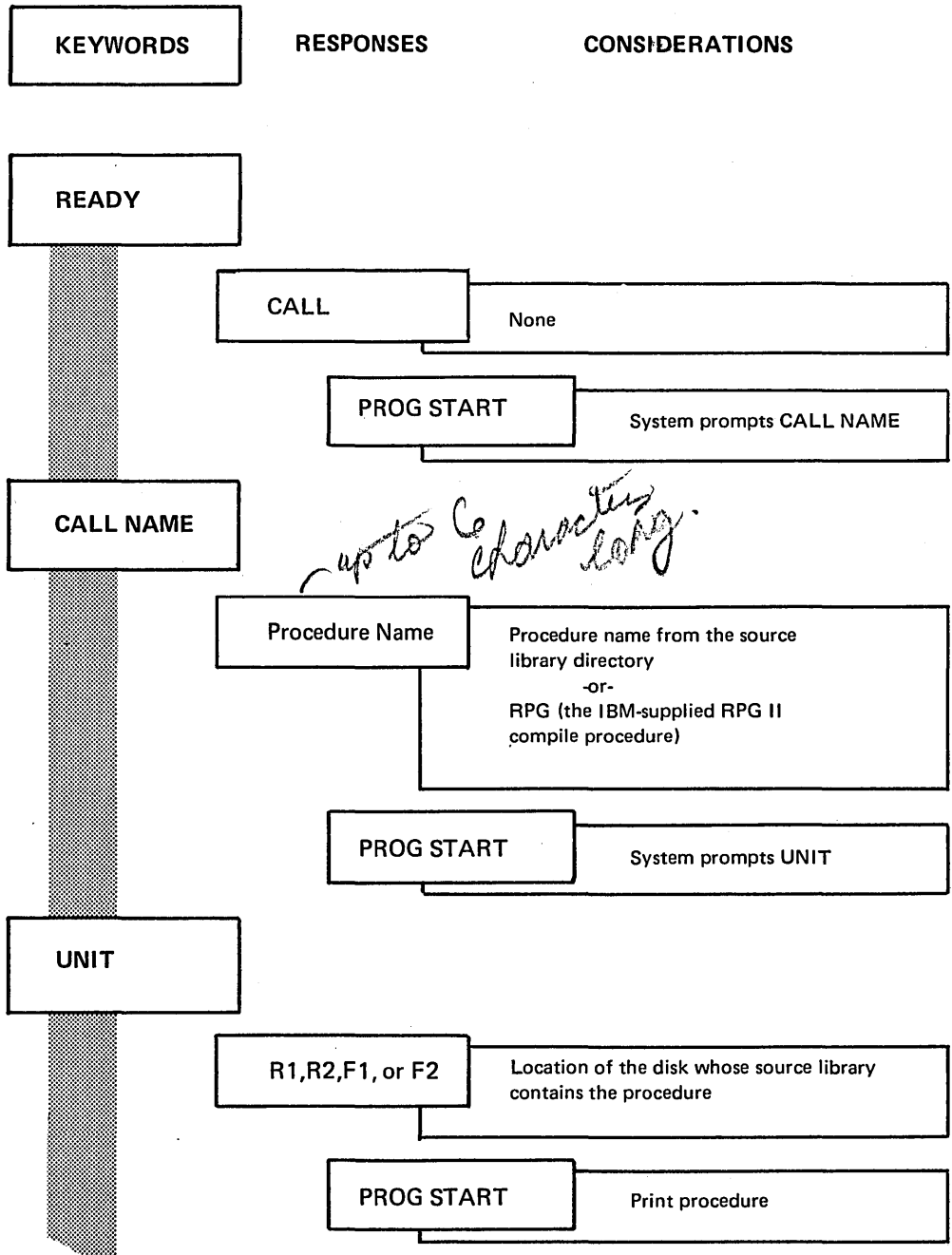
**INCLUDE**Add system program control statements  
to a procedure**PROG START**System prints 'ENTER INCLUDED  
STATEMENTS' and a 2-digit statement  
no**RUN**Tells system  
a. The BUILD cycle is complete.  
b. Run the job**PROG START**

System runs job

## Keyword Sequence for OCL Call Cycle



# Keyword-Response Summary (Call Cycle)



KEYWORDS

RESPONSES

CONSIDERATIONS

PROCEDURE DISPLAYED ON SYSTEM PRINTER <sup>①</sup>

MODIFY  
(Operator can use one, all, or a combination of the responses.)

LOG	Used only if CRT or 22" printer on system (see Appendixes D and E)
PROG START	System prompts LOG DEVICE
CANCEL	Cancel job
PROG START	System prompts READY or displays end-of-job halt.
FORMS	Change lines per page of printed output for system programs
PROG START	System prompts FORMS DEVICE
Asterisk (*) Followed by Comment	Enter comment
PROG START	System waits for next MODIFY response

① A.

Procedures with INCLUDE Statements

When a procedure contains SORT source statements or utility control statements, the display part of the CALL cycle is more complex. See *Considerations During a CALL Cycle*, under *MODIFY; Including Control Statements* in Part I.

B.

Procedures with Delayed Responses

The procedure is displayed statement by statement. When the system reaches a statement which contains a delayed response, it will display the statement keyword and wait for the operator's response.

KEYWORDS	RESPONSES	CONSIDERATIONS
	Statement number and comma	To delete statement in displayed procedure.
	PROG START	System waits for next MODIFY response.
	Statement number and corrected statement	To correct statement in displayed procedure.
	PROG START	System waits for correct statement
	RUN	Tells system – a. The CALL cycle is complete. b. Run the job
	PROG START	System runs job

**BUILD NAME**

When the system prompts BUILD NAME, the operator responds with a name for the procedure that will be put in a source library at the end of the sequence. (The operator's response to UNIT determines what source library the procedure will be put in.) At the end of the BUILD cycle, the system enters the procedure in the source library and puts the procedure name in the source library directory as a permanent entry.

Restrictions on naming a procedure are:

1. Name must not contain more than six alphanumeric characters. Blanks, commas, quotes (apostrophes), and periods are not allowed.
2. First character must be alphabetic (A-Z or #, @, \$).
3. Name must not be DIR, SYSTEM, or ALL (these names are reserved for system use).

**Duplicate Procedure Names**

If the operator's response to BUILD NAME duplicates the name of a procedure already in the source library directory, the system prints a message and reprompts BUILD NAME.

The operator can:

1. Proceed — by typing a different name or the same name and a different unit.
2. Proceed — by typing the same name and unit again. The new procedure will then overlay the old procedure in the source library.
3. End the job — see description of error message options in *IBM System/3 Model 6 Operator's Guide*, GC21-7501.

**Deleting a Source Library Procedure**

The system gives a P (permanent) designation to all procedures entered into a source library during a BUILD cycle. Therefore, the only way to delete a procedure from a source library is to run the Library Maintenance Utility Program. (For information about the Library Maintenance Utility Program see Part II of this manual.)

**BUILD DC NAME**

Refer to *Chained Procedures* in Appendix A.

**CALL NAME**

The response to CALL NAME is the name of the procedure you want to run. This can be either:

- The name of a procedure entered in a source library after a BUILD or BUILD DC cycle. (The operator's response to the keyword BUILD NAME, or BUILD DC NAME determines the name of the procedure.)
- RPG (the IBM-supplied RPG II Compile Procedure).

If the operator does not know the procedure name, he can get a printout of the source library directory by running the Library Maintenance Utility Program. (See Part II of this manual for more information about this program.)

## COMPILE OBJECT

The keyword COMPILE OBJECT requires a response (R1, R2, F1, or F2) if the system has more than one object library and you do not want to put the compiled RPG II program in the same object library where the RPG II Compiler resides.

If the operator does not respond to COMPILE OBJECT, but merely presses the PROG START key, the system places the compiled RPG II program in the object library where the RPG II Compiler resides.

- F1 refers to the fixed disk on drive one.
- R1 refers to the removable disk on drive one.
- F2 refers to the fixed disk on drive two.
- R2 refers to the removable disk on drive two.

## SOURCE

### In a LOAD Cycle

SOURCE is prompted only when the response to LOAD NAME is the name of a compiler (such as \$RPG). The response to SOURCE is the name of the source program you want to compile. (This name must be the one you used when you put the program in a source library during a KSE or Library Maintenance Program run. ①)

### In a BUILD Cycle

There are two possible responses to SOURCE during a BUILD cycle: the name of a source program you want to compile or a delayed response. Each response has a special significance to the system.

Response	Tells System
Name of Source Program You Want to Compile	You're building a procedure that will compile a particular source program. (The program must be in a source library.) The program name you supply must be the one you used when you put the program in a source library during a KSE or Library Maintenance Program run. ①
Delayed Response (?)	You're building a general RPG II compile procedure. You will supply the necessary source program information (name and location of the source program and where you want to put the compiled program) during the CALL cycle.

## UNIT; SOURCE UNIT

Possible responses to the keyword UNIT are F1, R1, F2, and R2.

- F1 refers to the fixed disk on drive one.
- R1 refers to the removable disk on drive one.
- F2 refers to the fixed disk on drive two.
- R2 refers to the removable disk on drive two.

---

① For information about the KSE Program see the *IBM System/3 Model 6 Conversational Utility Programs Reference Manual*, SC21-7528. For information about the Library Maintenance Program see PART II of this manual.



## **DATE (SYSTEM DATE)**

This DATE keyword lets the operator change the system date for a particular job. (The system date is used in headings on program listings, in headings on printed output, and in labels for new files.)

The system date is established at IPL time. This date is used for every job unless the operator overrides it.

### **Overriding the System Date**

The operator can override the system date for any single job by typing in a new date when the system prompts the keyword DATE. The new system date is used only for the one job. When that job is finished, the system date automatically reverts to its IPL setting.

### **Format of the DATE Statement**

Although the operator can override the system date, he cannot change the date format. The system date format is established during sysgen as either:

- mmddyy (month/day/year) – For U.S. installations
- ddmmyy (day/month/year) – For World Trade installations

The three elements (month/day/year) can be separated by any non-numeric symbol (except a comma, quotation mark, or blank) or run together without any separation.

In a system using the mmddyy format, for example, all of the following would be valid ways of typing May 12, 1971:

- 05/12/71
- 05-12-71
- 051271
- 5/12/71

## FILE NAME

For Each File Used in a Job, The Operator Must Supply This Type of Information:	By Responding to the Keyword:	With:
1. Name of File	FILE NAME	<ul style="list-style-type: none"> <li>● FILENAME from the file specification at compile time.</li> <li>● Predefined filename (for \$RPG, \$KDE, \$DSORT, \$COPY).</li> </ul>
2A. Location of disk where you want to write the file (For a file creation run) – or – 2B. Location of disk which contains the file to be processed (For all other runs)	UNIT	<ul style="list-style-type: none"> <li>● R1 or F1                (For systems with one disk drive)                – or –</li> <li>● R1, F1, R2, or F2                (For systems with two disk drives)</li> </ul>
3A. Name which identifies the disk on which you want to write the file (For a file creation run) – or – 3B. Name which identifies the disk on which the file is located (For all other runs)	PACK	Name assigned to disk by Disk Initialization Utility Program

### FILE NAME for Customer Programs

For a file used in an RPG II compiled customer program, the operator's response to FILE NAME is the name in columns 7-14 of the RPG II File Description Specifications.

### FILE NAME for \$RPG, \$DSORT, \$COPY, \$MICR, and \$KDE

For \$RPG's predefined file names see *IBM System/3 Model 6 RPG II Reference Manual, SC21-7517*.

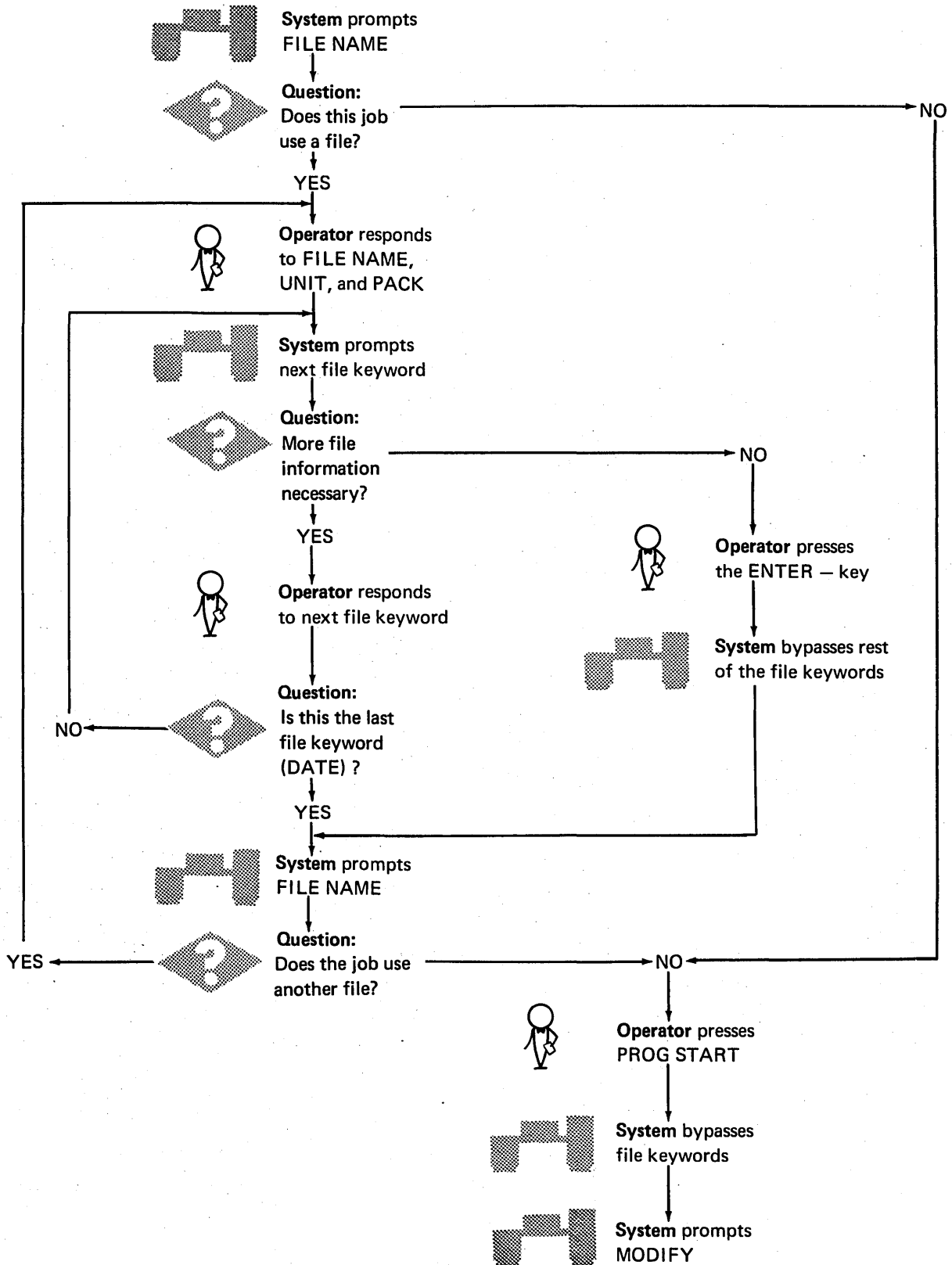
For \$DSORT see *IBM System/3 Disk Sort Reference Manual, SC21-7522*.

For \$COPY see Part II of this manual.

For \$MICR see *IBM System/3 Model 6 Utility Program for the IBM 1255 Magnetic Character Reader Reference Manual, SC21-7527*.

For \$KDE see *IBM System/3 Model 6 Conversational Utility Programs Reference Manual, SC21-7528*.

# System-Operator Interaction During Prompting of File Keywords



## Multiple Files

A job often involves several files. When this is the case, the operator must respond to several series of file keywords. The first time the system prompts the file keywords, the operator responds with information about one file. After the operator responds to DATE, the system will again prompt FILE NAME. This time the operator responds with the name of the second file.

When he has responded to the file keywords for all the files that will be used in the job, the operator should respond to FILE NAME by pressing PROG START. The system then bypasses the rest of the file keywords and prompts MODIFY.

A maximum of 15 file statements can be used for each job.

## UNIT; FILE UNIT

Possible responses to the keyword UNIT are F1, R1, F2, and R2.

- F1 refers to the fixed disk on drive one.
- R1 refers to the removable disk on drive one.
- F2 refers to the fixed disk on drive two.
- R2 refers to the removable disk on drive two.

## PACK

Whenever a job involves a disk file you must tell the system the name of the disk where the file is (or will be) located, so the system can make sure that disk is mounted before the job is begun. To tell the system the name of the disk the file is on, the operator responds to the keyword PACK with the name assigned to the disk during its initialization. (The Disk Initialization section of Part II of this manual explains the procedure for naming a new disk.)

Although most installations keep a record of the names and contents of each of their disk packs, the operator can always get the name of any disk by running the File and Volume Label Display Utility Program. The disk name is part of the output of this program.

## LABEL

When a file is created, the system enters a file name in the VTOC. The keyword LABEL refers to this VTOC file name. Unless the operator responds to LABEL, the name entered in the VTOC is the same as the operator's response to FILE NAME.

LABEL requires a response:

1. At file creation time, if you want the VTOC file name to be different from the operator's response to FILE NAME. (For example, if the RPG II file name is A but the disk already has an A file, a response to LABEL would be required, and the response would have to be something other than A.)
2. During a program run, if you are using a file whose VTOC file name is different from the operator's response to FILE NAME.

## RECORDS (AND TRACKS)

When a file is created, the operator must tell the system how much disk space to allocate for the file. He does this by responding to one of the two space keywords: TRACKS and RECORDS. (If the operator responds to RECORDS, TRACKS will not be prompted.)

The following chart shows the possible responses to these keywords and how the system interprets the responses.

Keyword	Operator Response	Tells System
TRACKS	1-398	Number of disk tracks needed for the file
RECORDS	1-999999	Number of records in the file

## Responding to TRACKS

The response to TRACKS is the number of disk tracks the records in a file will occupy. (Appendix B reviews how to convert the number of records in a sequential, direct, or indexed file into the number of tracks that would be required to contain the file records on a disk.)

## Responding to RECORDS

If the operator does not want to convert record numbers into track requirements himself, the system will do it for him. The system determines the track requirements for a file when the operator responds to RECORDS.

## LOCATION

LOCATION requires a response during file creation if you want to control the placement of files on the disk. LOCATION is required when creating several versions of the same file. It can also be used to reference one of several files having the same name.

The response to LOCATION is the track where you want the file to begin. Possible responses are 8 through 405. (Tracks 0 through 7 are reserved for system use.)

If the operator does not respond to the keyword LOCATION when a new file is created, the system places the file in whatever available area it fits best.

## RETAIN

The keyword RETAIN applies to file designation. Files can be designated: P (permanent), T (temporary), or S (scratch).

The operator responds to RETAIN either:

1. At file creation, to give a designation to the file being created.
2. When accessing a file, to change the designation of a file from T to S or from S to T.

## File Creation

A file designation (along with the file name, length, and other related information) is placed in the VTOC when a file is created. The operator controls file designation by his response to RETAIN. (If the operator does not respond to RETAIN, the system gives the file a T designation.)

## Permanent Files

Because permanent files are protected against inadvertent overlaying or altering, give a P designation to all the files you want to keep.

## Temporary Files

Give a T designation to a file if you plan to use it several times within a couple of days and will not need it after that.

## Scratch Files

Give an S designation to any file you plan to use only once. When a scratch (S) 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 VTOC is to change a T entry to an S by using RETAIN-S in the file statement, or change a T or P entry to S by using a \$DELET SCRATCH statement.

The file designation dictates how much freedom you have in overlaying or changing a file. The following chart summarizes how each file designation restricts your freedom to overlay or change a file.

## File Designations

## Restrictions

P

The only way to change a permanent file is to delete it by running the File Delete Utility program.

T

The only way to overlay a temporary file is to load a new file over it. To do this, the operator's responses to all the file keywords must duplicate those of the present T file.

S

The system will overlay a scratch file if the disk pack is full and/or file space is needed.

## Changing File Designation of Existing File

When the system prompts RETAIN, the operator can:

- Accept the current file designation. (By pressing PROG START)
- Change a temporary file to a scratch file (by typing an S). The VTOC will contain an S entry for the file.
- Change a scratch file listed in the VTOC to a temporary file by typing an A.

## Deleting Files

The operator can delete any file by running the File Delete Utility Program, which changes the file designation in the VTOC to S. This effectively deletes the entire file, because the system will overlay the file area as soon as more file space is needed. When the file area is overlaid, the file name is erased from the VTOC.

## DATE (FILE DATE)

This keyword (prompted after the keyword RETAIN) refers to the system date in effect when a file was created.

The system date is established at IPL. This date is used for every job unless the operator overrides it.

DATE requires a response only if the job being run uses a file whose name and label are duplicated by another file on the same disk. In this case, the operator responds to DATE by typing in the system date in effect when the file he wants to use was created. With this date, the system can distinguish one file from others on the same disk with the same VTOC file name and label.

If neither the date nor the location is given, the file having the latest date is the one automatically referenced.

If the operator does not know what the system date was when the file was created, he can get a printout of the creation dates for all files on a disk by running the File and Volume Label Display Utility Program. (Detailed information on this program is available in Part II of this manual.)

## Restriction During File Creation

A response to DATE tells the system that this file already exists. If DATE is entered during a file creation run a FILE NOT FOUND error occurs.

## HALT

The operator can respond to the keyword READY with HALT. The system will then halt at the end of each job. HALT need only be entered to cancel the effect of a NOHALT statement.

## LOAD NAME

### For Customer Programs

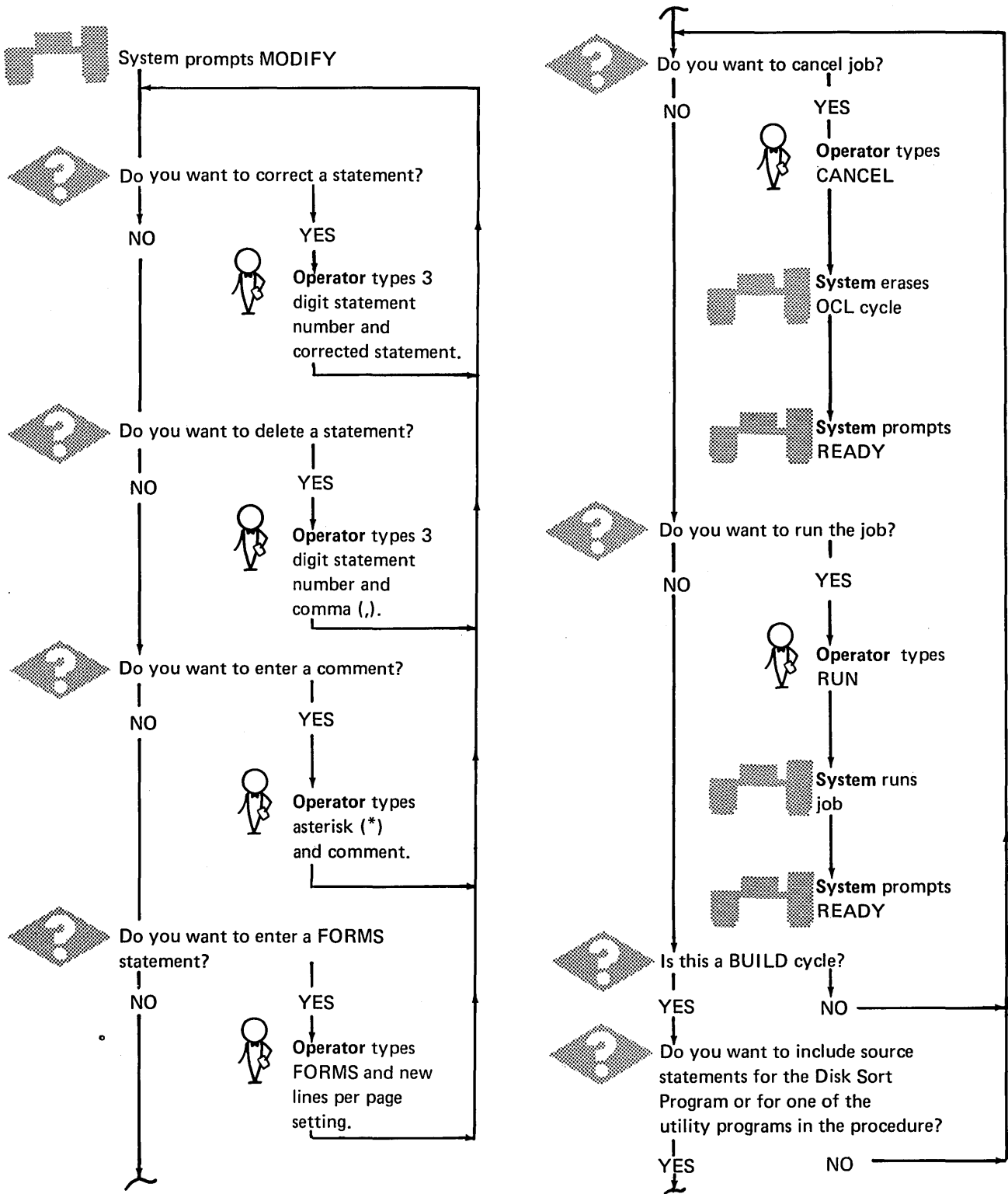
The response to LOAD NAME is the name of the customer's RPG II program.

### For System Programs

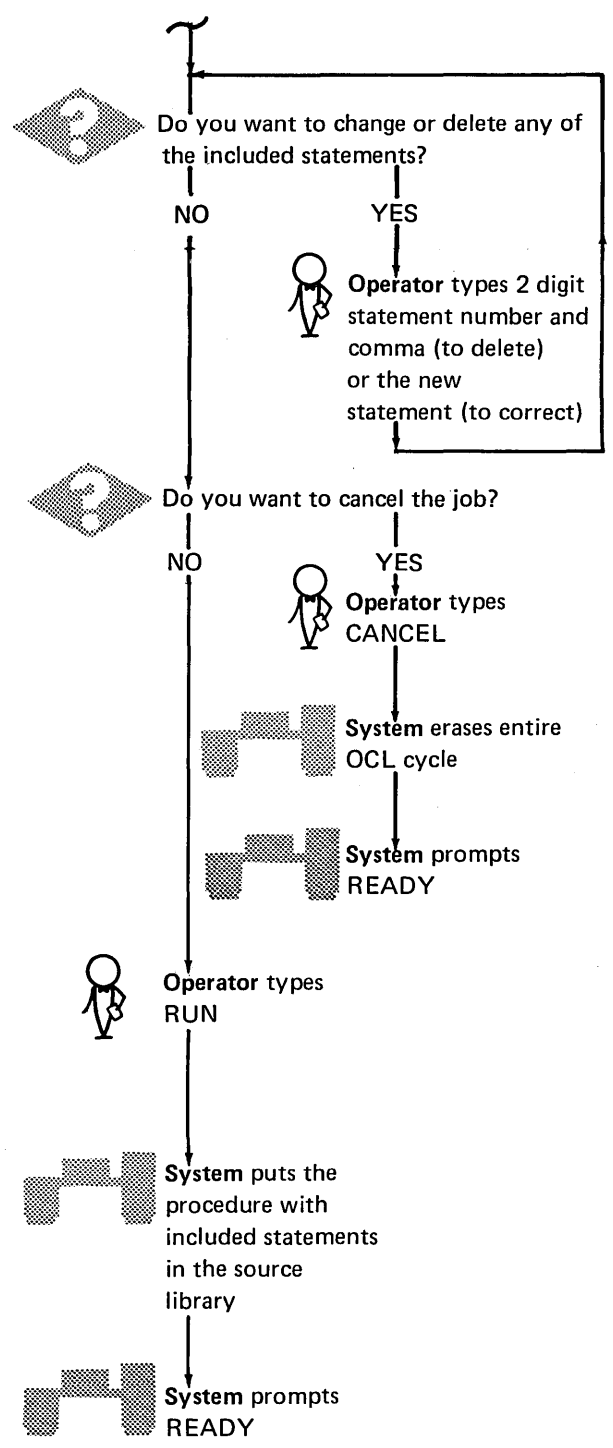
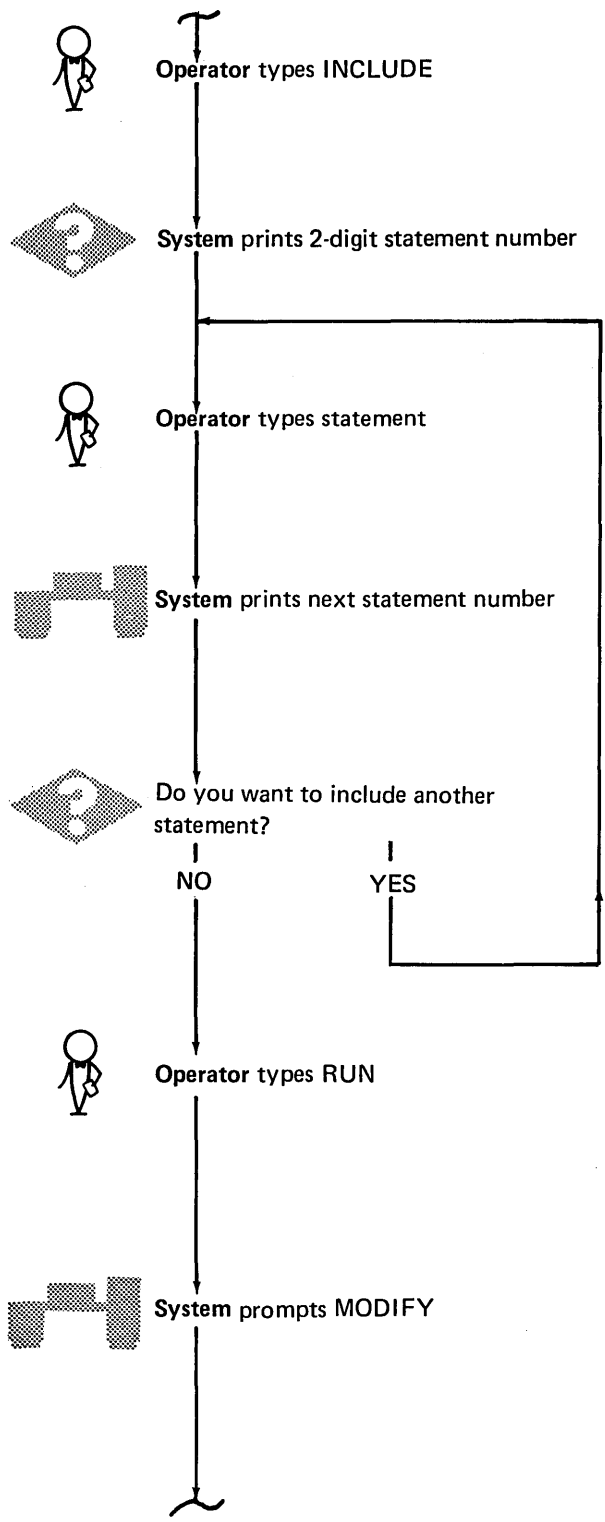
The response to LOAD NAME is the name of the specific system program you want to run.

Name	Program	More Information About the Program In
\$ALT	Alternate Track Assignment	Part II of this manual
\$BUILD	Alternate Track Rebuild	
\$COPY	Disk Copy/Dump	
\$INIT	Disk Initialization	
\$LABEL	File and Volume Label Display	
\$DELET	File Delete	
\$MAINT	Library Maintenance	
\$KSE	Keyboard Source Entry	<i>IBM System/3 Model 6 Conversational Utilities Reference Manual, SC21-7528</i>
\$KDE	Keyboard Data Entry	
\$DIU	Data Interchange	
\$MICR	1255 Magnetic Character Reader Utility	<i>IBM System/3 Model 6 Utility Program for 1255 Magnetic Character Reader Reference Manual, SC21-7527</i>
\$RPG	RPG II Compiler	<i>IBM System/3 Model 6 RPG II Reference Manual, SC21-7517</i>
\$DSORT	Disk Sort	<i>IBM System/3 Disk Sort Reference Manual, SC21-7522</i>

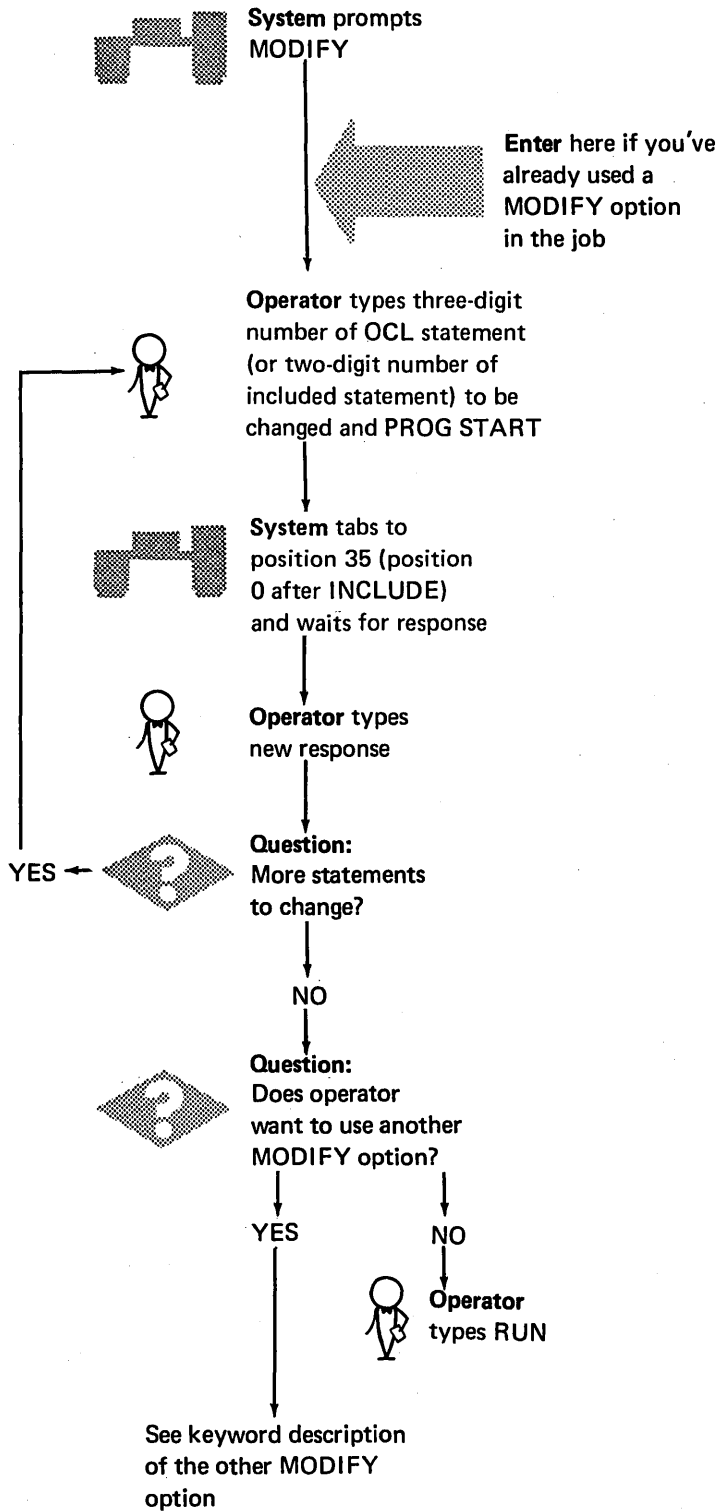
# MODIFY



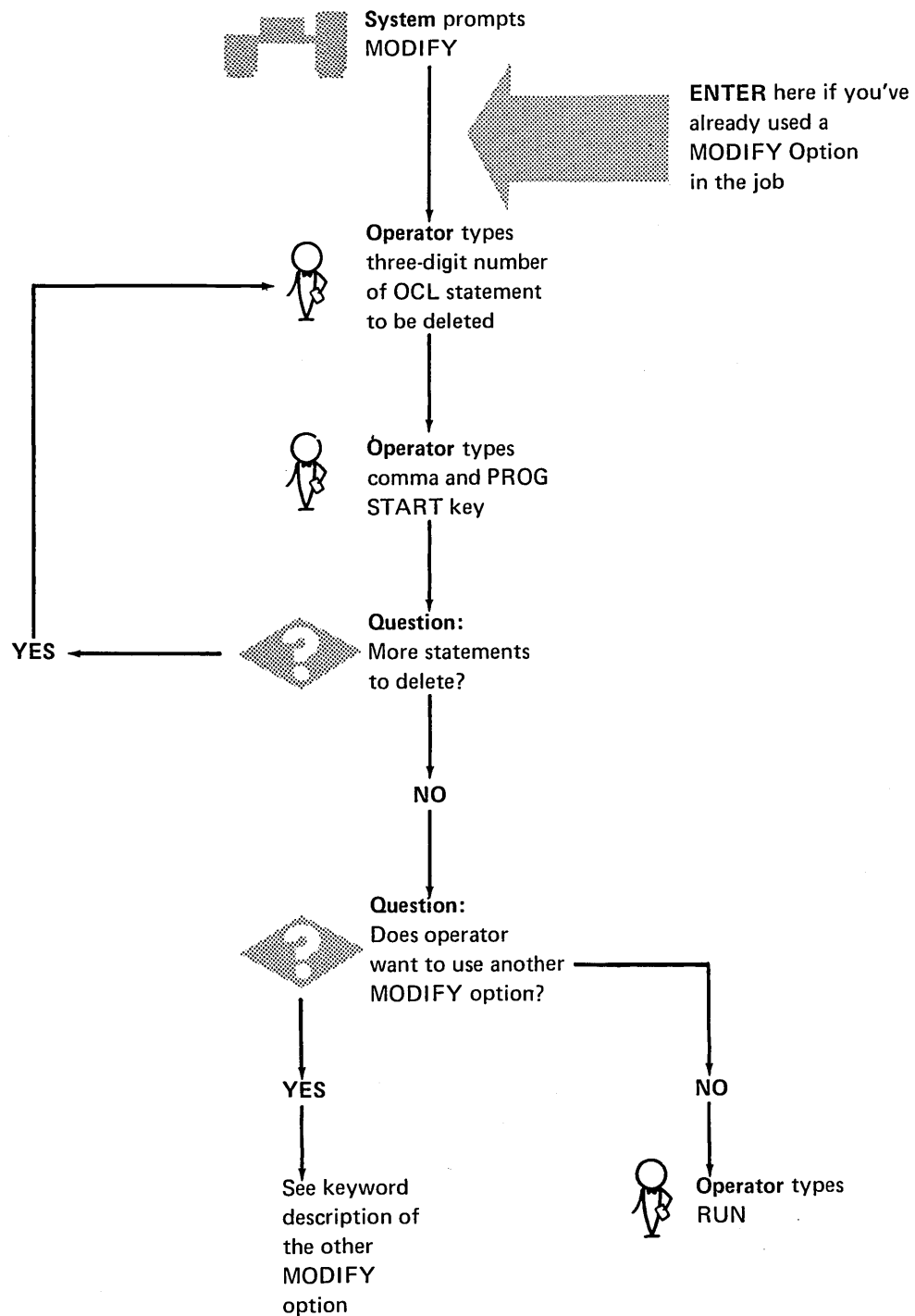




# MODIFY; Changing a Previous OCL Statement



## MODIFY; Deleting a Previous OCL Statement

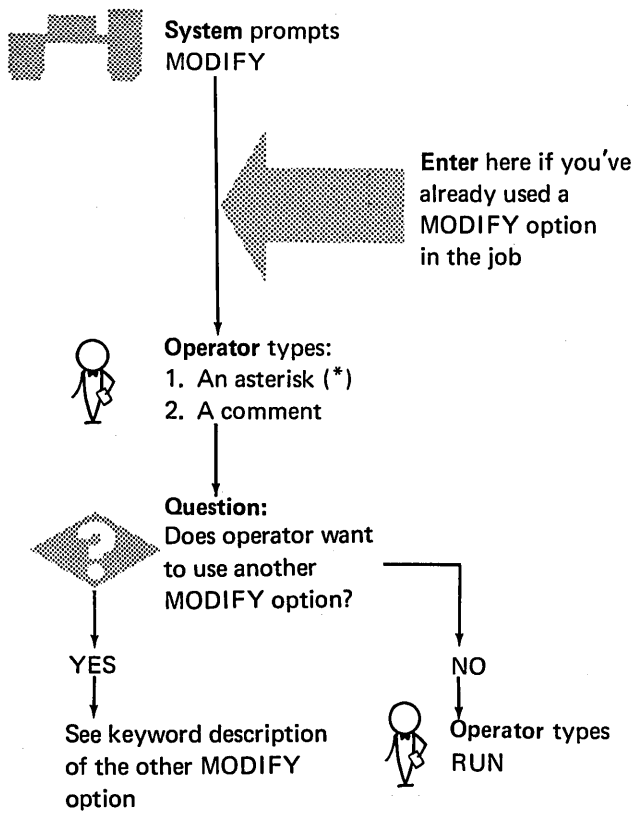


### Deleting Multiple Keywords

When the OCL statement number for **FILE NAME** is deleted, all keywords for that file will be deleted from the cycle. For example, the **LABEL** or **DATE**

keywords could be deleted from a file keyword statement without deleting the other keywords for that file. However, if **FILE NAME** is deleted, that entire file would be deleted from the cycle.

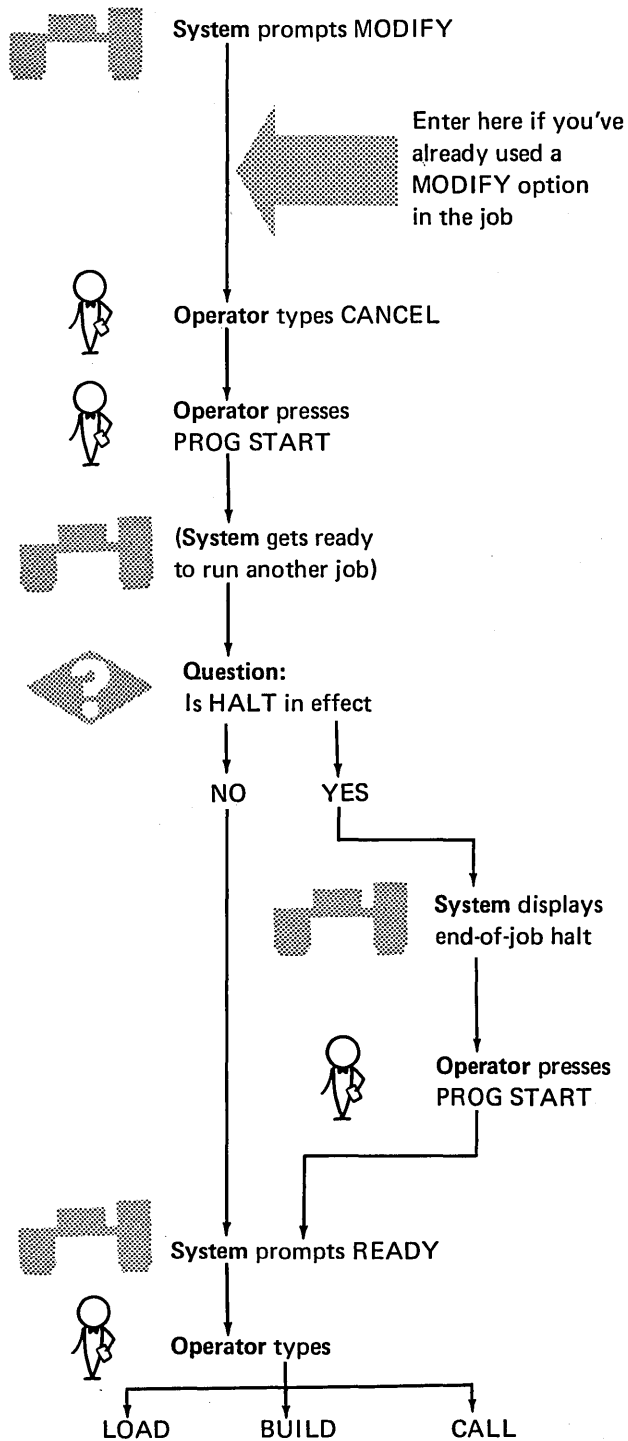
## MODIFY; Entering Comments



## Points to Remember When Entering Comments

- The usual purpose of a comment is to remind the operator of something he must do (mount a new disk pack, for example) or to document a problem during a program run.
- After the operator types a comment, it is immediately displayed on the system printer.
- Comments typed during a BUILD cycle become a permanent part of the procedure. They are entered into the Source Library along with OCL statements.
- Comments typed during a LOAD or CALL cycle do not become a permanent part of the job; their only purpose is to help document the program run.

## MODIFY; Cancelling Job



### Effect of Entering CANCEL During a LOAD Cycle

The entire LOAD cycle will be overlaid by the next OCL cycle.

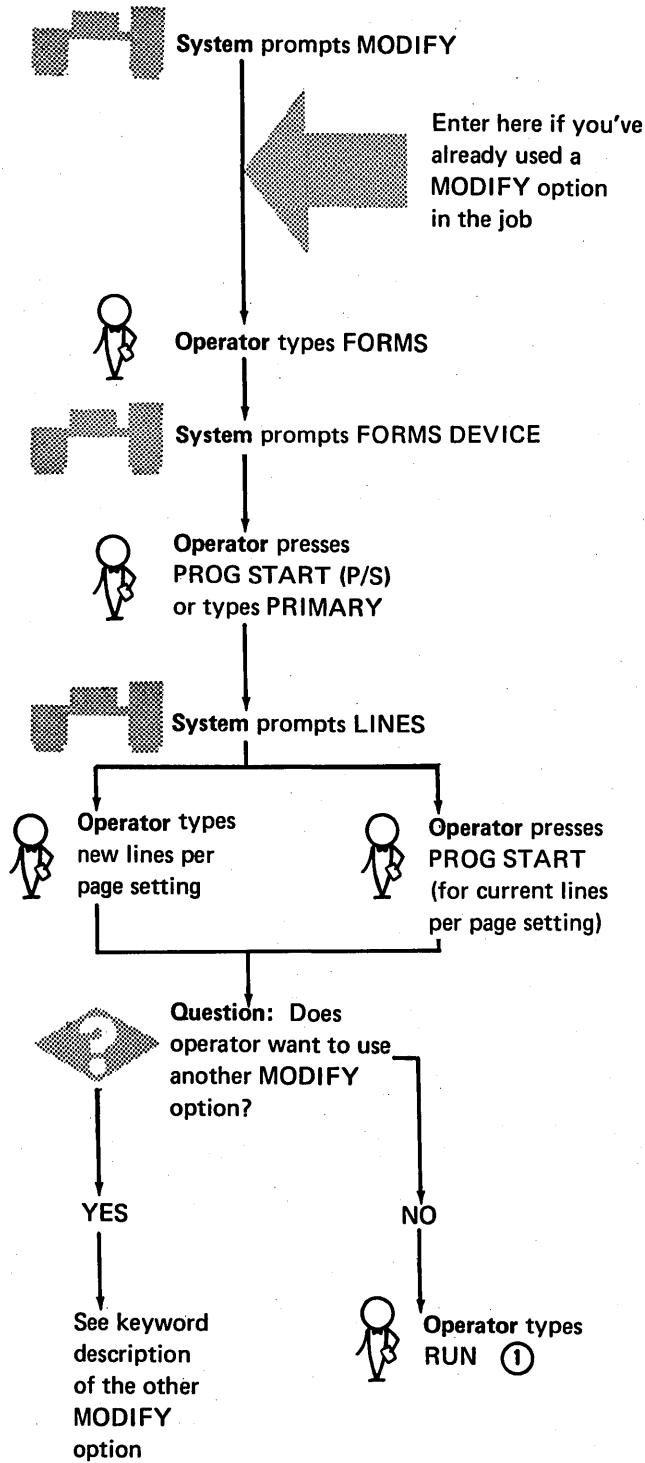
### Effect of Entering CANCEL During a BUILD Cycle

The entire BUILD cycle will be overlaid by the next OCL cycle. (If a duplicate procedure is being built, and CANCEL entered, the original procedure remains in the source library. Except: if CANCEL is entered after INCLUDE, neither procedure will be in the library.)

### Effect of Entering CANCEL During CALL Cycle

The entire CALL cycle will be overlaid by the next OCL cycle. The original procedure will be unchanged.

## MODIFY; Entering Forms



① Whenever the keyword FORMS is entered in an OCL sequence a system halt occurs after RUN in case the operator needs to change the paper in the printer. The system remains idle until the operator enters zero and presses the PROG START key.

### **Purpose of FORMS**

Standard outputs for Model 6 printers is 66 lines per page. At IPL time, 66 lines per page is established as the forms length unless a different value was specified during system generation.

To change the lines per page of printed output for RPG II programs, you code line counter specifications. To change the lines per page of printed output for system programs (utilities, SORT, and the RPG Compiler), you type the keyword FORMS and an appropriate response.

If line counter specifications and an OCL FORMS statement are both used in one job, and if the specified lengths are different, the system will accept the RPG II line counter specifications and ignore the OCL FORMS statement.

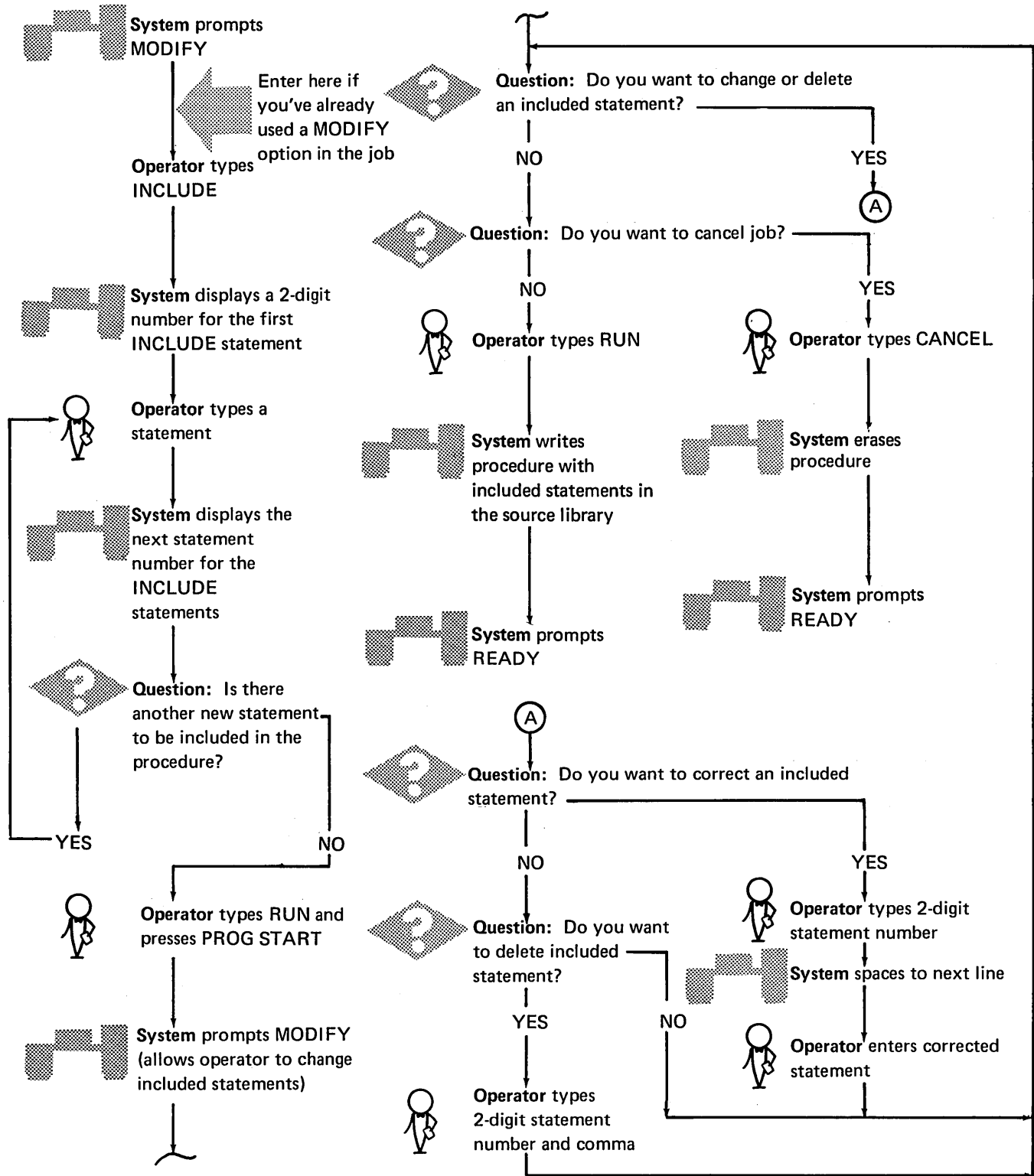
The new lines per page setting (from either an OCL FORMS statement or an RPG II line counter specification) remains effective until another OCL FORMS statement or RPG II line counter specification is read.

FORMS can be entered during the MODIFY phase of any OCL cycle. (The system never prompts FORMS.)

Whenever the operator types FORMS during an OCL cycle, a system halt follows RUN in case the operator needs to change the paper in the printer. Job processing does not resume until the operator enters a zero (option 0) and presses the PROG START key.

For additional operating information, including line counter considerations, related to the keyword FORMS, see the *IBM System/3 Model 6 Operator's Guide*, GC21-7501

# MODIFY; Including Control Statements





### Purpose of INCLUDE

The keyword INCLUDE lets you add system program control statements to a procedure. INCLUDE tells the system that the next entry will be a set of control statements for one of the system programs. (As used here, control statements refer to both the control statements for the utility programs and the sequence specifications for the SORT program.) A maximum of 25 control statements can be included in each procedure.

### Restrictions After INCLUDE

After including statements in a procedure, the procedure cannot be changed. MODIFY is prompted to allow changing included statements. If CANCEL is used after INCLUDE in a procedure that overlaid a duplicate procedure, neither the original nor the new procedure will be in the source library.

### Considerations During a CALL Cycle

When the operator uses the CALL cycle to get the procedure out of the source library, the system displays the procedure in two separate steps: first the OCL statements, then the INCLUDE statements. The following shows details of the two display steps:

1. System displays OCL statements for the job.
  2. System prompts MODIFY (to give operator a chance to correct any of the OCL statements).
  3. Operator, after he has made any necessary corrections, types RUN.
  4. System displays heading: INCLUDED STATEMENTS.
  5. System prints the INCLUDE statements.
  6. System prompts MODIFY (to give operator a chance to correct any of the INCLUDE statements).
  7. Operator, after he has made any necessary corrections, types RUN.
  8. Model 6 runs the job.
- 
- System displays OCL statements
- System displays INCLUDE statements

### NOHALT

Normally the system halts when a job ends. The operator can respond to the keyword READY with NOHALT. The system will then prompt READY for the next job when each job ends. The NOHALT will remain in effect until a HALT statement is entered or an IPL occurs.

## READY

When the system is ready to begin the OCL sequence for a new job, it prompts READY.

The operator responds by typing the name of one of the four OCL cycles: LOAD, BUILD, BUILD, or CALL. The system then prompts the other key-words in the sequence.

(OCL cycles for the Model 6 are described in the *Summary of Conversational OCL* at the front of this manual.)

## RUN

RUN is the last entry in any OCL cycle. The operator types RUN when he is satisfied that the OCL cycle is complete and correct. The table shows what happens when the operator types RUN during any of the three OCL cycles.

Sequence	Effect
LOAD	Job is run.
CALL ①	
BUILD ①	The OCL statements are put in a source library.
① If INCLUDE statements are part of the procedure the BUILD and CALL cycles require two RUN entries. (See <i>Considerations During a CALL Cycle</i> under <i>MODIFY - Including Control Statements</i> in Part I.)	

After the operator types RUN, the system processes the job and end-of-job occurs.

The system then prompts READY for the next job.

## SWITCH

The OCL SWITCH statement allows changing the eight external indicators used by RPG II programs.

(External indicators are discussed in the *IBM System/3 Model 6 RPG II Reference Manual; SC21-7517.*)

The operator-system interaction involved with the SWITCH statement is different for each OCL cycle as shown in the following charts.

### Indicator Settings

The indicator setting has eight positions, corresponding to the eight external indicators.

The three possible entries for each position are:

- 1 – sets corresponding indicator on.
- 0 – sets the corresponding indicator off.
- X – leaves the corresponding indicator unchanged.

For example, if the operator keys in XXXX10XX:

- Indicator five will be set on.
- Indicator six will be set off.
- Indicators one, two, three, four, seven, and eight will be unchanged.

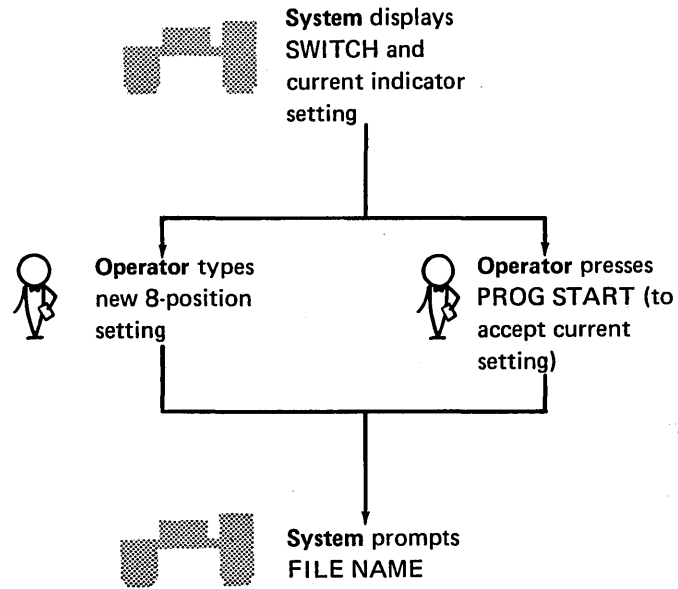
### IPL Considerations

All eight external indicators are set off at IPL. The only way to set an indicator on is by responding to the keyword SWITCH with a new eight-position response containing a 1 in the appropriate position.

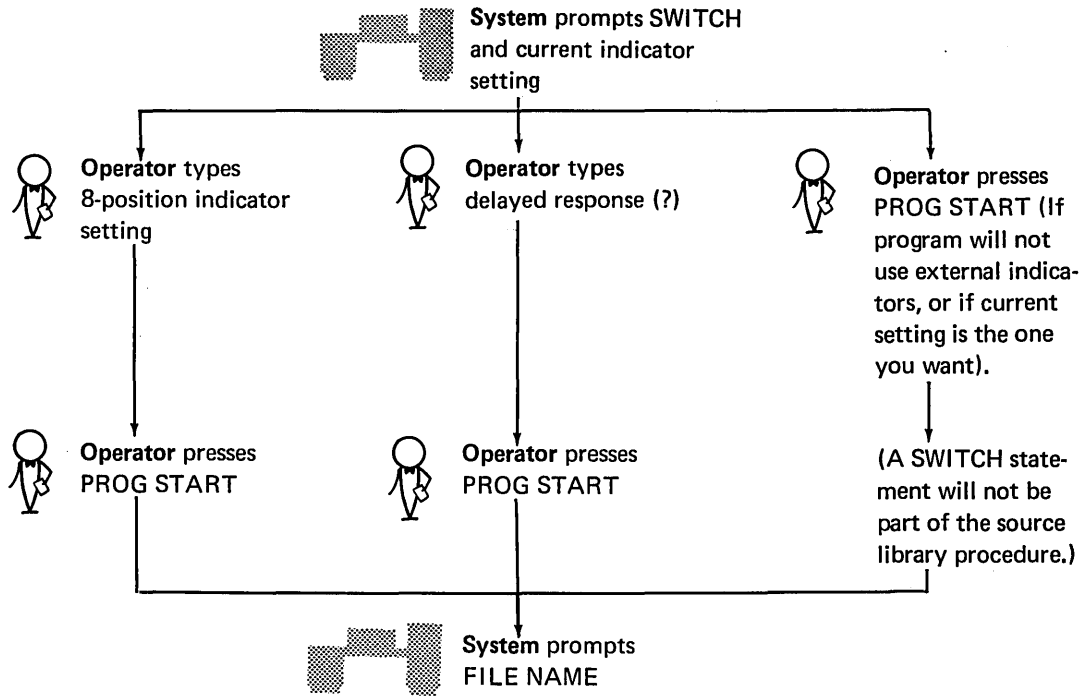
### Duration of SWITCH Setting

When an OCL SWITCH statement sets an indicator on, the indicator remains on until another SWITCH statement sets it off or the next IPL occurs.

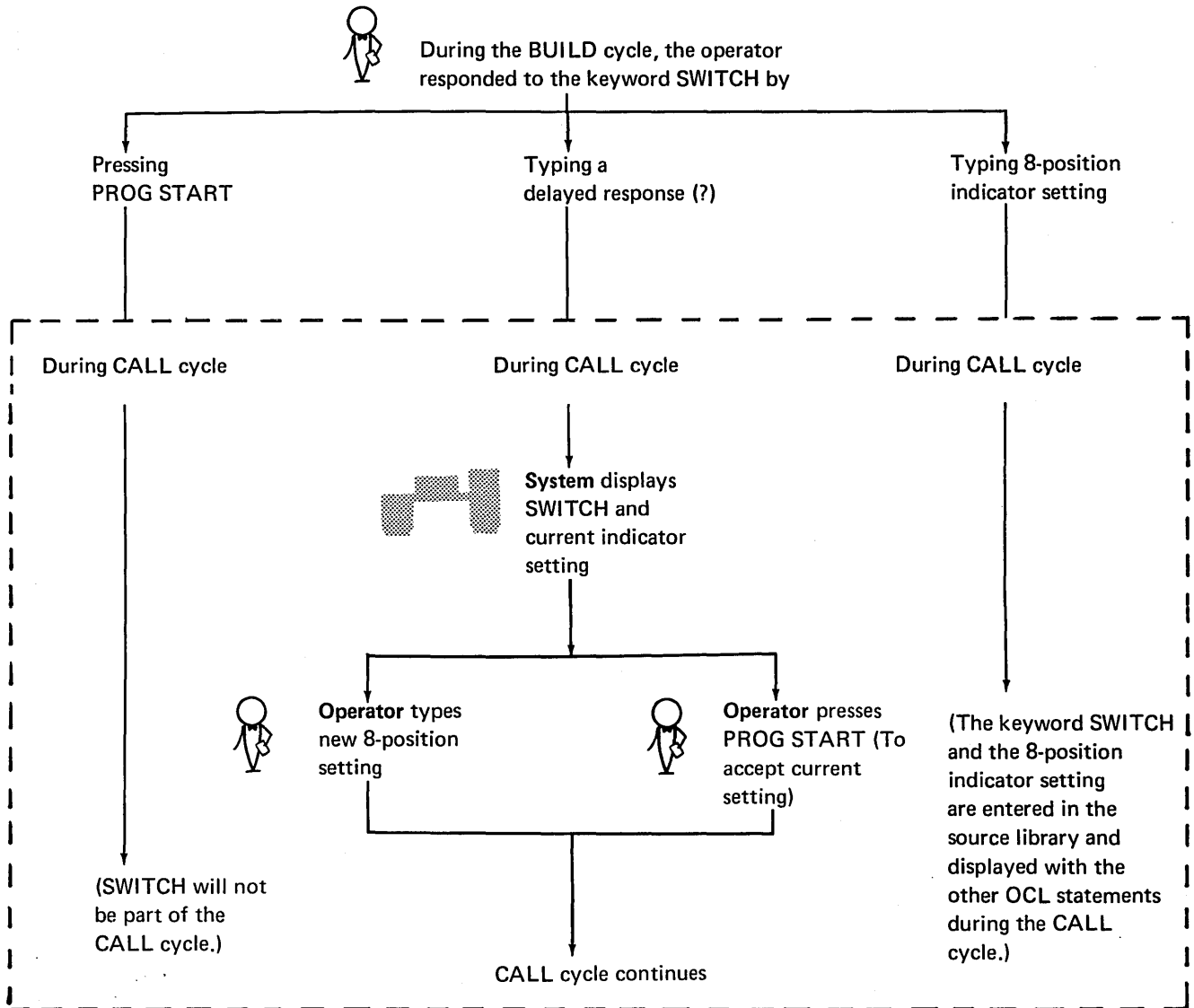
# Operator-System Interaction For SWITCH Statement (LOAD Cycle)



# Operator-System Interaction For SWITCH Statement (BUILD Cycle)



# Operator-System Interaction for SWITCH Statement (CALL Cycle)





This section presents a sequence of six typical jobs:

1. Initialize a disk.
2. Compile an RPG II source program.
3. Run the compiled program.
4. Copy a file from one disk to another.
5. Build a procedure to run a multi-file job.
6. Call and modify the procedure built in job 5.

Each sample job is organized into three sections:

1. An introductory summary explaining the job.
2. The OCL statements (and – where applicable – the utility control statements) for the job.
3. Explanatory notes on individual statements in the job.

The examples shown are actual computer printouts. End-of-statement keys used are shown in parenthesis to indicate actual operator response. These are shown for example only and will not be printed on normal OCL printouts.

Any response without end-of-statement key indicated is printed by the system without operator intervention.

## SAMPLE JOB 1. INITIALIZE DISK

We're going to use the Disk Initialization Program (located on the fixed disk on drive one) to initialize the removable disk on drive one. We want to:

- Initialize the entire disk pack:
- Do surface analysis only once.

The name of the new disk will be 12345.

Here are the OCL and utility control statements for the job.

```
READY-                                LOAD (P/S)
*****
010 LOAD                               $INIT (P/S)
011                                     F1 (ENTER-)
*****
MODIFY

RUN (P/S)
  ENTER '// ' CONTROL STATEMENT
// UIN UNIT-R1,TYPE-PRIMARY (P/S)
  ENTER '// ' CONTROL STATEMENT
// VOL PACK-12345 (P/S)
  ENTER '// ' CONTROL STATEMENT
// END (P/S)
```



## Explanation

- 010 LOAD NAME – \$INIT  
\$INIT is the system name for the Disk Initialization Program.
- 011 UNIT – F1 The Disk Initialization Program is located on the fixed disk on drive one. Pressing ENTER- instead of PROG START to end response causes DATE, SWITCH, and File keywords to be bypassed.
- // UIN UNIT – R1, TYPE-PRIMARY
  1. Tells the system to initialize the removable disk on drive one.
  2. Because no other parameters are entered in the UIN statement, the program will:
    - o Initialize the entire pack.
    - o Read and verify the test data on the pack one time.
- // VOL PACK–  
12345 – \$INIT will enter the disk name 12345 in the VTOC. Whenever a file from this disk is used in a job, the operator must type 12345 when the system prompts PACK.
- // END End of control statements.

## SAMPLE JOB 2. COMPILE AN RPG SOURCE PROGRAM

We're going to use the IBM-supplied procedure RPGB (located in the source library on the fixed disk on drive one) to compile a source program INVUPD (an inventory update) located on R1. The RPG II Compiler (the program to compile RPG II source programs) is also located on R1. We want to put the compiled program in the object library on R1. Here are the OCL statements for the job.

```
READY-
000 CALL          NAME--          CALL (P/S)
001              UNIT--          RPGB (P/S)
001              UNIT--          F1 (P/S)
*****
010 LOAD          NAME-$RPG
011              UNIT-R1
020 COMPILE      OBJECT-F1
021              SOURCE-          INVUPD (P/S)
022              UNIT-R1
030 FILE         NAME-$WORK
031              UNIT-F1
032              PACK-F1F1F1
033              TRACKS-20
034              RETAIN-S
040 FILE         NAME-$SOURCE
041              UNIT-F1
042              PACK-F1F1F1
043              TRACKS-20
044              RETAIN-S
*****
MODIFY
020 (P/S)          R1 (P/S)
RUN (P/S)
```

## Explanation

- 000 CALL NAME      – RPGB  
Tells the system you want to use the IBM supplied Compile Procedure (RPGB).
  
- 010 LOAD NAME      – \$RPG  
Tells the system you want to use the RPG II Compiler (the program to compile RPG II source programs).
  
- 011 UNIT            – R1  
The RPG II Compiler is located on R1.
  
- 020 COMPILE  
  OBJECT              – F1  
The object program will be put in the object library of the disk on F1.
  
- 021 SOURCE         – INVUPD  
The SOURCE statement in the RPGB procedure requires a delayed response. When the system reaches the SOURCE statement in the display sequence, it prompts SOURCE and waits for the operator's response.
  
- 022 UNIT            – R1  
The response tells the system that the program to be compiled (INVUPD) is located on R1.
  
- 020 MODIFY         – R1
  1. System prompts MODIFY.
  2. Operator types 020, telling system he wants to change that statement. (He does not want the system to put the compiled program on F1.)
  3. System tabs to position 37 and waits for response.
  4. Operator types new response—R1. The system will put the compiled program on R1.

### SAMPLE JOB 3. PROCESS CUSTOMER PROGRAM "INVUPD"

We're going to run the customer program INVUPD, compiled in SAMPLE JOB 2 and located on the removable disk on drive one. The job uses one file, INV, located on R2. The name of the disk which contains the file INV is 123456. Here are the OCL statements for the job.

```
READY-                                LOAD (P/S)
*****
010 LOAD          NAME-                INVUPD (P/S)
011              UNIT-                R1 (P/S)
020 DATE    (12/08/70) -              (P/S)
030 SWITCH (00000000) -              (P/S)
040 FILE          NAME-                INV (P/S)
041              UNIT-                R2 (P/S)
042              PACK-                123456 (P/S)
043              LABEL-               (ENTER-)
050 FILE          NAME-                (P/S)
*****
MODIFY

RUN (P/S)
```

## Explanation

- 020 DATE – (12/08/70)  
We'll use the current system date for the job.
- 030 SWITCH – (00000000) – (P/S)  
The program doesn't use external indicators so the operator doesn't care about the switch setting and responds by pressing the PROG START key.
- 043 LABEL – Press the ENTER- key  
Responding to LABEL by pressing the ENTER- key tells the system to bypass the rest of the file keywords and prompt FILE NAME.
- 050 FILE NAME – (P/S)  
Responding to FILE NAME by pressing PROG START causes the system to bypass the rest of the file keywords and prompt MODIFY.

#### SAMPLE JOB 4. COPY FILE DISK TO DISK

We're going to copy an employee master file from R1 to R2. The second file will serve as a back-up in case the original file is damaged in some way, such as a track becoming defective or a portion of the file being overlayed. When the master file was created the programmer:

1. Responded to FILE NAME with EMASTFIL.
2. Responded to PACK with VOL06.
3. Responded to LABEL with EMPMAST.
4. Responded to TRACKS with 15.

These responses caused the system to put the name EMPMAST in the VTOC on VOL06.

Here are the OCL and utility control statements we will use to copy the master file from R1 to R2.

```
READY--                                LOAD (P/S)
*****
010 LOAD          NAME--                $COPY (P/S)
011              UNIT--                F1 (P/S)
020 DATE    (12/08/70) -                (P/S)
030 SWITCH (00000000) -                (P/S)
040 FILE          NAME--                COPYIN (P/S)
041              UNIT--                R1 (P/S)
042              PACK--                VOL06 (P/S)
043              LABEL--                EMPMAST (ENTER--)
050 FILE          NAME--                COPYO (P/S)
051              UNIT--                R2 (P/S)
052              PACK--                VOL07 (P/S)
053              LABEL--                EMPMAST2 (P/S)
054              RECORDS--              (P/S)
055              TRACKS--               15 (P/S)
056              LOCATION--             (P/S)
057              RETAIN--               P (ENTER--)
060 FILE          NAME--                (P/S)
*****
MODIFY

RUN (P/S)
  ENTER '// ' CONTROL STATEMENT
// COPYFILE OUTPUT-DISK (P/S)
  ENTER '// ' CONTROL STATEMENT
// END (P/S)
```

## Explanation

- 010 LOAD NAME – \$COPY  
\$COPY is the system name for the Disk Copy/Dump Program.
- 011 UNIT – F1  
The Copy Disk Program is on F1.
- 020 DATE – (12/08/70)  
We'll use the current system date for the job.
- 030 SWITCH – (00000000)  
This program doesn't use external indicators, so operator doesn't care about the switch setting and responds by pressing PROG START.
- 040 FILE NAME – COPYIN  
COPYIN is the predefined file name you must use for the input file whenever you use Disk Copy/Dump Program.
- 043 LABEL – EMPMAST  
EMPMAST is the VTOC file name for the COPYIN file. You must supply this name so the system knows which file to use for COPYIN. Pressing the ENTER- key causes the system to bypass the rest of the file keywords and prompt FILE NAME.
- 050 FILE NAME – COPYO  
COPYO is the predefined file name you must use for the output file whenever you use the Disk Copy/Dump Program.
- 053 LABEL – EMPMAST2  
The system enters EMPMAST2 in the VTOC on VOL07. EMPMAST2 is the name by which the system will identify the back-up file.
- 055 TRACKS – 15  
Because we are creating a new file we must respond to one of the space keywords (TRACKS and RECORDS). We specify 15 tracks because that's what we specified for the original file.
- 057 RETAIN – P  
The back-up file is to be permanent to protect it against inadvertent overlaying. Pressing the ENTER- key causes the system to bypass the rest of the file keywords and prompt FILE NAME.
- COPYFILE OUTPUT – DISK  
The COPYFILE statement tells the program to copy the designated file from R1 to R2.

## SAMPLE JOB 5. MULTI-FILE BUILD

Each day the customer runs a daily transaction job which creates a daily transaction file. Each day's file has a different name and date. We are going to build a procedure to use these daily files to create a weekly transaction file (WKLYTR). The weekly transaction program is located in the object library of fixed disk 1.

```

READY-
000 BUILD          NAME-          BUILD (P/S)
001                UNIT-          WTR (P/S)
                                R2 (P/S)
*****
010 LOAD          NAME-          WKYRUN (P/S)
011                UNIT-          F1 (P/S)
020 DATE          -              (P/S)
030 SWITCH (00000000) -          (P/S)
040 FILE          NAME-          MONTR          MONDAYS FILE (P/S)
041                UNIT-          F1 (P/S)
042                PACK-          PACK08 (P/S)
043                LABEL-          (P/S)
044                RECORDS-          (P/S)
045                TRACKS-          (P/S)
046                LOCATION-          (P/S)
047                RETAIN-          (P/S)
048                DATE-          ? (P/S)
050 FILE          NAME-          TUETR          TUESDAYS FILE (P/S)
051                UNIT-          F1 (P/S)
052                PACK-          PACK08 (P/S)
053                LABEL-          (P/S)
054                RECORDS-          (P/S)
055                TRACKS-          (P/S)
056                LOCATION-          (P/S)
057                RETAIN-          (P/S)
058                DATE-          ? (P/S)
060 FILE          NAME-          WEDTR          WEDNESDAYS FILE (P/S)
061                UNIT-          F1 (P/S)
062                PACK-          PACK08 (P/S)
063                LABEL-          (P/S)
064                RECORDS-          (P/S)
065                TRACKS-          (P/S)
066                LOCATION-          (P/S)
067                RETAIN-          (P/S)
068                DATE-          ? (P/S)

```



070 FILE	NAME-	THUTR	THURSDAYS FILE (P/S)
071	UNIT-	F1 (P/S)	
072	PACK-	PACK08 (P/S)	
073	LABEL-	(P/S)	
074	RECORDS-	(P/S)	
075	TRACKS-	(P/S)	
076	LOCATION-	(P/S)	
077	RETAIN-	(P/S)	
078	DATE-	? (P/S)	
080 FILE	NAME-	FRITR	FRIDAYS FILE (P/S)
081	UNIT-	F1 (P/S)	
082	PACK-	PACK08 (P/S)	
083	LABEL-	(P/S)	
084	RECORDS-	(P/S)	
085	TRACKS-	(P/S)	
086	LOCATION-	(P/S)	
087	RETAIN-	(P/S)	
088	DATE-	? (P/S)	
090 FILE	NAME-	WKLYTR (P/S)	
091	UNIT-	R1 (P/S)	
092	PACK-	PACK04 (P/S)	
093	LABEL-	(P/S)	
094	RECORDS-	500 (P/S)	
095	LOCATION-	(P/S)	
098	RETAIN-	P (ENTER-)	
100 FILE	NAME-	(P/S)	

\*\*\*\*\*

MODIFY

RUN (P/S)

## Explanation

- 000 BUILD NAME – WTR  
The procedure name in the source library is WTR.
- 001 UNIT – R2  
The procedure is located on unit R2.
- 020 DATE – (P/S)  
The date statement is not part of the procedure.
- 030 SWITCH – (00000000) – (P/S)  
The external indicators are not used by the program.
- 040 FILE NAME – MONTR MONDAYS FILE  
The file name for each day is different. The comment (MONDAYS FILE) will become part of the procedure.
- 048 DATE – ? (P/S)  
The date each file was created is supplied at CALL time, when the job is run.
- 090 FILE NAME – WKLYTR (P/S)  
The output file is called WKLYTR and put on PACK04 on unit R1.
- 094 RECORDS – 500 (P/S)  
Our output file contains up to 500 records.
- 096 RETAIN – P (ENTER-)  
We want to make this a permanent file. The ENTER- key caused DATE to be skipped and FILE NAME prompted.
- 100 FILE NAME – (P/S)  
We are finished with file statements, prompt MODIFY.
- RUN – Put the procedure in the source library.

## SAMPLE JOB 6. MULTI-FILE CALL

We are going to run the procedure we built in sample job 5. However, this week Thursday was a holiday so there are only four input files. We can still use the same procedure if we delete an input file at MODIFY time.

```

READY-
000 CALL          NAME--          CALL (P/S)
001              UNIT--          WTR (P/S)
                                R2 (P/S)
*****
010 LOAD          NAME--WKYRUN
011              UNIT--F1
020 FILE          NAME--MONTR
021              UNIT--F1
022              PACK--PACK08
023              DATE--          4/5/71 (P/S)
030 FILE          NAME--TUETR
031              UNIT--F1
032              PACK--PACK08
033              DATE--          4/6/71 (P/S)
040 FILE          NAME--WEDTR
041              UNIT--F1
042              PACK--PACK08
043              DATE--          4/7/71 (P/S)
050 FILE          NAME--THUTR
051              UNIT--F1
052              PACK--PACK08
053              DATE--          4/8/71 (P/S)
060 FILE          NAME--FRITR
061              UNIT--F1
062              PACK--PACK08
063              DATE--          4/9/71 (P/S)
070 FILE          NAME--WKLYTR
071              UNIT--R1
072              PACK--PACK04
073              RECORDS--500
074              RETAIN--P

```

```

*****
MODIFY

```

050, (P/S)

\* THURSDAYS FILE DELETED BECAUSE OF HOLIDAY, NO RUN THAT DAY (P/S)

RUN (P/S)

## Explanation

- 023 DATE                   – 4/5/71
- 033 DATE                   – 4/6/71
- 043 DATE                   – 4/7/71
- 053 DATE                   – 4/8/71
- 063 DATE                   – 4/9/71  
    We must supply the date for each day's input file because we gave a delayed response (?) at BUILD time. Thursday's date is entered even though we will delete the file later. A date should be entered to continue the cycle.
  
- MODIFY 050               – We delete the entire file for Thursday and enter a comment to explain why.
  
- RUN                       – Start the job.



**PART II.  
DISK UTILITY PROGRAMS**





To use utility programs, you must write utility control statements and operation control language (OCL) statements. In this manual, therefore, the information for every program is divided into five sections:

- Control statement summary
- Parameter summary
- Parameter descriptions
- OCL considerations
- Examples

The first three sections are to guide you in writing utility control statements. The OCL section is to guide you in writing OCL statements. The examples will help you in both.

### Writing Utility Control Statements

You may write utility control statements on whatever paper or preprinted forms you like. In writing the statements, use the manual in the following way:

1. Look at the **CONTROL STATEMENT SUMMARY** to determine which control statements and parameters apply to the program use you are interested in. (The program uses are stated in the text preceding the control statement summary.)
2. If you need information about the contents or meanings of particular parameters, look at the **PARAMETER SUMMARY**.
3. If you need more detailed information about parameters, read the **PARAMETER DESCRIPTIONS** following the parameter summary.
4. If you need examples of specific jobs, look at the **EXAMPLE** section. All examples show the OCL and utility control statements needed to load and run the utility programs for specific jobs. The statements are shown in the form they are printed on the system printer.

### Writing OCL Statements

To write OCL statements to run a utility program, look at the **OCL CONSIDERATIONS** section for that program. There you will find a list of the required keywords and responses for **LOAD** and **BUILD** sequences. (Keywords not listed can be bypassed.) Should you need more general information about OCL, or more specific information about the keywords, see Part I of this manual.

### 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 this manual.

### Utility Control Statements

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 you can use are listed in the parameter summaries for the control statements.

Braces ( { } ) sometimes appear in parameters shown in control 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**.

### OCL Statements

In OCL statements, keywords are capitalized. Responses that are shown in capital letters must be written as shown. If numbers or special characters are included with the capital letters, they must be written as part of the response. For example, **\$INIT** is the name of the Disk Initialization program and must be written exactly as shown. Responses that are not capitalized mean you must use the value that applies to the job you are doing.





The disk system management programs include the following utility programs:

- Disk Initialization
- Alternate Track Assignment
- Alternate Track Rebuild
- File and Volume Label Display
- File Delete
- Disk Copy/Dump
- Library Maintenance

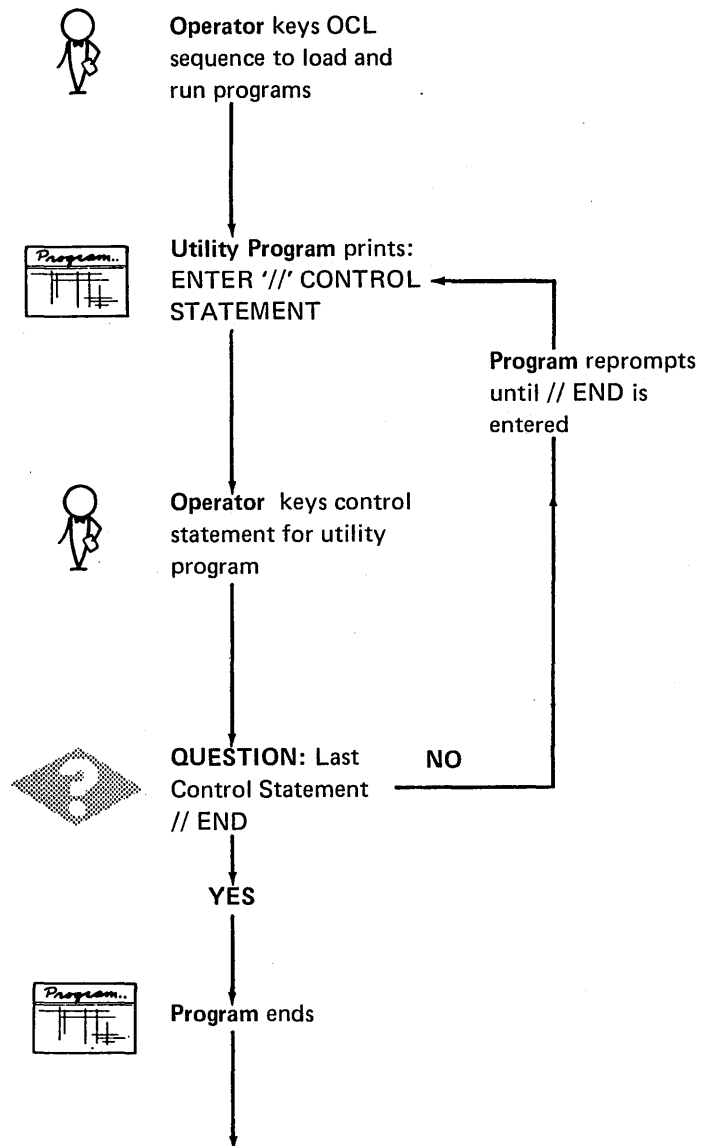
These programs, residing on disk, do a variety of necessary jobs: from preparing disks for use to adding new or changed programs to the system.

**General Program Operation**

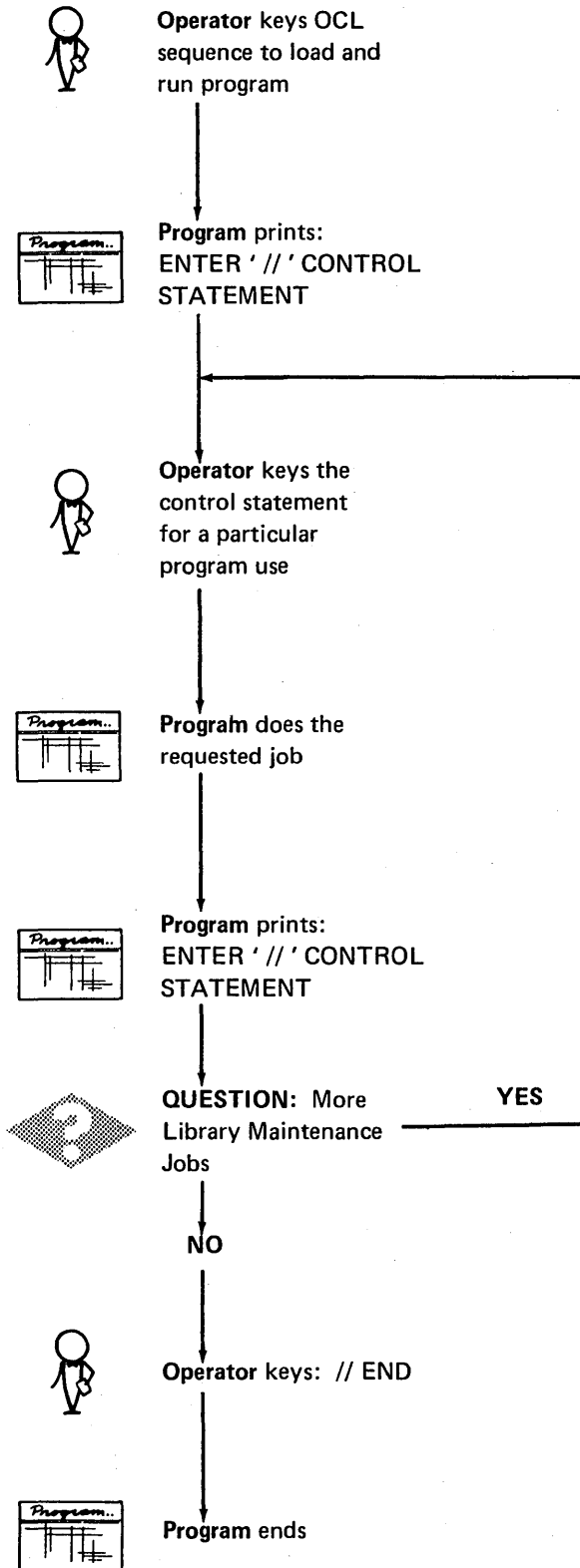
The utility programs require control statements describing the jobs you want done. They read these statements from the system input device, or from procedures stored in a source library on disk. The system input device is normally the keyboard, but the operator can specify another device by his response to the OCL keyword READER during initial program loading (IPL).

The following diagrams outline the general way the utility programs operate. Assume that the programs are reading control statements from the keyboard.

**All Programs Except Library Maintenance**



## Library Maintenance Program



## Control Statements

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 (following // blank in positions 1-3). 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 writing control statements are as follows:

1. *//blank.* All control statements must have // blank in positions 1-3.
2. *Statement Identifier.* Begin it in position 4 or after of the statement. Do not use blanks within the identifier.
3. *Blanks.* Use one or more blanks between the identifier and the first parameter. Do not use them anywhere else in the statement.
4. *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.
5. *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).
6. *Statement length.* Control statements must not exceed 96 characters.

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

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

**End Control Statement**

The END statement is a special control statement that indicates the end of control statements. It consists of the letters // END in positions 1-6 and must always be the last control statement for the programs.



The Disk Initialization program (\$INIT) prepares disks for use. It does this by:

- Writing track and sector addresses on the disk.
- Checking for defective tracks, a process called surface analysis.
- Assigning alternate tracks to any defective tracks found.
- Writing a name on each disk to identify the disk.
- Formatting cylinder 0 (zero).

The process is called initialization. The program can initialize up to three disks during the same program run.

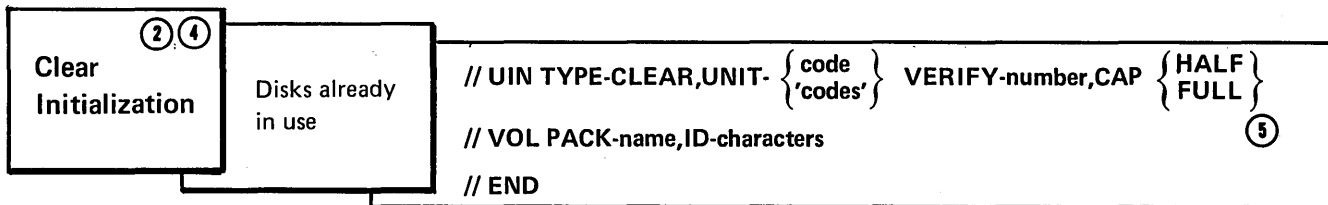
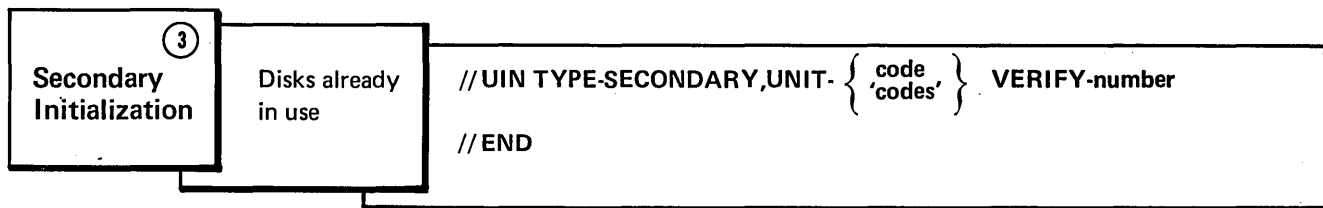
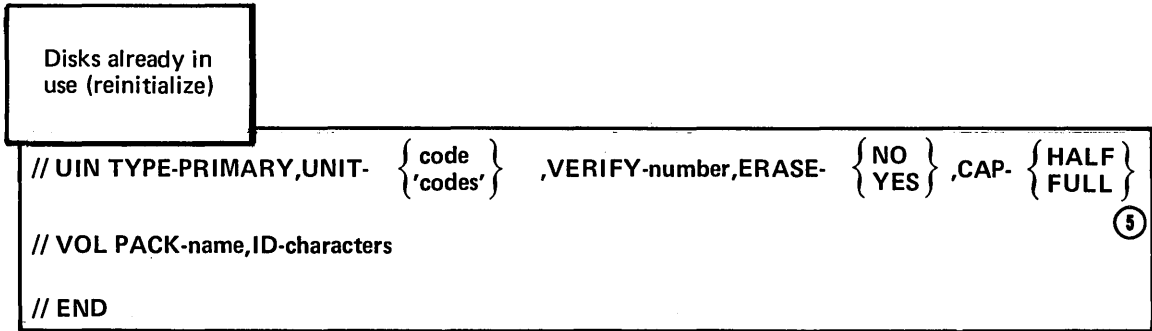
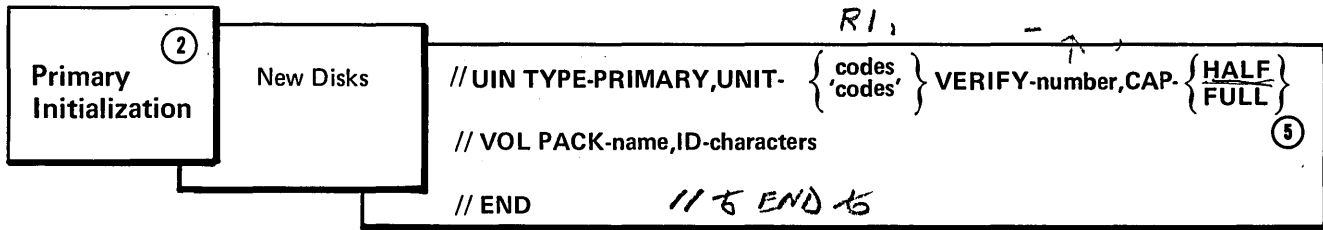
All disks must be initialized before use. Disks that have been initialized need not be re-initialized unless you want to erase their contents and rename them.

There are three types of initialization: primary, secondary, and clear. Primary is used to initialize the entire disk. Secondary is used only when the disk capacity of your system is increased and you have programs and data on your disks that you want to keep. Clear is used to unconditionally initialize a disk regardless of the presence of any files or libraries. Therefore, the use of this parameter is not recommended. 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 for \$INIT

— Use —

— Control Statements —



- ① Control statements are required in the order they are listed: UIN, VOL, END or UIN, END. The TYPE-PRIMARY, VERIFY, and ERASE parameters are optional.
- ② For primary or clear initialization, 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, and so on.
- ③ VOL statements are not required for secondary initialization because the disks are already named.
- ④ If the TYPE parameter CLEAR is selected, ERASE-YES is assumed.
- ⑤ CAP-FULL should not be used on a half capacity system.

# PARAMETER SUMMARY

## UIN (Input Definition) Statement

**TYPE-PRIMARY**

Primary initialization. Initialize the disks to the capacity of the drives on which they are mounted. Tracks already initialized are reinitialized.

**TYPE-SECONDARY**

Secondary initialization. Applies only to disks that were initialized on drives of less capacity than the drives you are now using. It means initialize the uninitialized portions of the disks to the capacity of the drives on which the disks are mounted. Tracks already initialized are not disturbed.

**TYPE-CLEAR**

Clear initialization. Initialize the disks to the capacity of the drives on which they are mounted. Tracks already initialized are reinitialized. Active files and library checking is bypassed and any data on the tracks is destroyed. Error logging areas on F1 are saved.

**UNIT-code**

Disk location (one disk).

**UNIT-code,code'**

Disk location (two disks).

**UNIT-code,code,code'**

Disk location (three disks).

Possible codes are R1,F1,R2,F2.

**VERIFY-number**

Do surface analysis the number of times indicated (number can be 1-255). VERIFY-1 is assumed if you omit the parameter.

**ERASE-YES**

Retest defective tracks.

**ERASE-NO**

Do not retest defective tracks.

Primary initialization only. ERASE-NO is assumed if you omit the parameter.

**CAP-HALF**

Initialize a disk to half capacity even if on a full capacity drive.

**CAP-FULL**

Initialize a disk to full capacity.

The CAP keyword forces ERASE-YES. Pack is initialized to the capacity of the drive if this keyword is omitted.

## VOL (Volume) Statement

PACK-name

Disk name. Can contain any of the standard System/3 characters except apostrophes (') and leading or embedded blanks. Its length must not exceed six characters.

ID-characters

Additional identification. Can contain any of the standard System/3 characters except apostrophes (') and leading or embedded blanks. Its length must not exceed ten characters. If you omit this parameter, no additional identification is written on the disk.

## PARAMETER DESCRIPTIONS

### TYPE Parameter (UIN)

The TYPE parameter indicates the type of initialization you want the program to do: primary, secondary, or clear. The type of initialization and the capacity of the disk drives on which the disks are mounted determine which disk tracks will be initialized. If this parameter is omitted, primary is assumed.

### Disk Drive Capacity

Disk Drives of different data-storage capacities are available for System/3 Model 6. All drives use the same type of disks. The only difference is the number of tracks the drives can use: the larger the drive capacity, the more tracks the drive can use. However, you must initialize the disk tracks before using them.

### Primary Initialization

Primary initialization applies to new disks, or disks you have used but want to initialize again. The program initializes all tracks corresponding to the capacity of the drives on which the disks are mounted. 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 libraries, temporary data files, or permanent data files. You must delete data files with the File Delete Program and libraries with the allocate function of the Library Maintenance Program.

### Secondary Initialization

Secondary initialization applies to disks that were initialized on drives of less capacity than drives you are now using. When you increase the capacity of your drives, more tracks on your disks become available for use. You must initialize the additional tracks. Use secondary initialization if you do not want information destroyed on tracks already in use. The program initializes the additional tracks only. Tracks already in use are not disturbed.

The program will not do secondary initialization on new disks or disks that have already been initialized to the capacity of the drives on which they are mounted.

### Clear Initialization

Clear initialization applies to new disks but only to those which cannot be used because of invalid pack labels or some other unrecoverable disk error. All tracks corresponding to the capacity of the drives on which the disks are mounted are initialized. Tracks that were previously initialized are reinitialized.

*Warning:* All libraries, temporary data files, or permanent data files are completely wiped out.

### UNIT Parameter (UIN)

The UNIT parameter (UNIT-code) tells the location of the disks you want to initialize. The program can initialize up to three disks during one program run.



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

1. For one disk, use UNIT-code
2. For two disks, use UNIT-'code,code'
3. For three disks, use UNIT-'code,code,code'

The codes indicate the locations of the disks:

Code	Location
R1	Removable disk on drive 1.
F1	Fixed disk on drive 1.
R2	Removable disk on drive 2.
F2	Fixed disk on drive 2.

For primary and clear initialization, the order of codes must correspond to the order of VOL control statements. If, for example, you had used the parameter UNIT-'R1,R2', the first VOL statement applies to the removable disk on drive 1 and the second VOL statement to the removable disk on drive 2. (No VOL statements are required for secondary initialization. The disk is already named.)

### 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 1-255. If this parameter is omitted, VERIFY-1 is assumed.

### 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 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 parameter, surface analysis is done once. Tracks that cause reading or writing errors any time during surface analysis are considered defective, but can be assigned alternates.

If the program finds more than six defective tracks, it considers the disk unusable and stops initializing the disk. Only six alternate tracks are available. (If you specified ERASE-NO, try to reinitialize with ERASE-YES.)

If either track 0 or 1 is defective, the program considers the disk unusable and stops initializing it. Tracks 0 and 1 are used only by the system and cannot have alternates assigned to them.

### ERASE Parameter (UIN)

The ERASE parameter concerns alternate track assignment. It applies only to disks that have already been initialized and used, but you are reinitializing using primary initialization.

The condition of tracks on such disks has been tested at least once before (during the previous initialization) and tracks that were found to be defective during surface analysis were assigned alternates. The ERASE parameter, therefore, enables 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.

The parameter ERASE-YES means to retest. If you tell the program to retest, it erases any existing alternate track assignments, and tests all tracks as though the disk were new.

The parameter ERASE-NO means not to retest. If you tell the program not to retest, it tests only those tracks to which no alternate tracks are assigned. Alternate tracks previously assigned remain assigned.

### CAP Parameter

The CAP parameter determines the size of the pack when it is initialized. The CAP-HALF parameter means to initialize the pack to half capacity even if it is on a full capacity drive. The CAP-FULL parameter means to initialize the pack to full capacity. The use of the CAP keyword forces ERASE-YES.

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

### **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. Any time a program attempts to use the defective track, it will automatically use the alternate instead. Each disk has six alternate tracks (tracks 2-7).

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

### **PACK Parameter (VOL)**

The PACK parameter (PACK-name) applies to primary and clear initialization only. During primary and clear 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 (') and leading or embedded blanks (see Appendix J). Its length must not exceed six characters. The following are valid disk names: 0, F0001, 012, A1B9, ABC.

In general, disk names are used for checking purposes. 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, a message to the operator is printed. 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 primary and clear initialization only. It enables you to include up to ten characters, in addition to the disk name, to further identify a disk. The information is strictly for your use. (It is not used for checking purposes by the system.) If you use the File and Volume Label Display program to print the disk name, it will also print the additional identification for you.

The additional identification can be any combination of standard System/3 characters except apostrophes (') and leading or embedded blanks. However, the maximum number is ten.

## OCL CONSIDERATIONS

LOAD Sequence		
<i>Keywords</i>	<i>Responses</i>	<i>Considerations</i>
READY	LOAD	-----
LOAD NAME	\$INIT	Name of Disk Initialization program.
UNIT	R1, R2, F1, or F2	Location of disk containing Disk Initialization program.
MODIFY	RUN	-----
↑ Only the key-words listed here are required. You can bypass the rest.	↑ You end every response by pressing PROG START.	

BUILD Sequence		
<i>Keywords</i>	<i>Responses</i>	<i>Considerations</i>
READY	BUILD	-----
BUILD NAME	Procedure name	Name by which procedure will be identified in source library.
UNIT	R1, R2, F1, or F2	Location of disk containing source library.
LOAD NAME	\$INIT	Name of Disk Initialization program.
UNIT	R1, R2, F1, or F2	Location of disk containing Disk Initialization program.
MODIFY	<ul style="list-style-type: none"> <li>• INCLUDE utility control statements</li> <li>RUN</li> <li>• RUN</li> </ul>	Response when including control statements in procedure.  Response when not including control statements in procedure.
↑ Only the key-words listed here are required. You can bypass the rest.	↑ You end every response by pressing PROG START.	

## EXAMPLE

### Primary Initialization of Two Disks

```

READY          - LOAD
*****
010   LOAD     NAME     - $INIT
011           UNIT     - F1
020   DATE     (XX/XX/XX) -
030   SWITCH   (00000000) -
040   FILE     NAME     -
*****

```

MODIFY

RUN

#### OCL LOAD Sequence

Circled areas are operator responses.

Keywords for which no responses are shown are the ones bypassed. If you press ENTER— after responding to UNIT, the DATE, SWITCH, and FILE NAME keywords are not prompted.

RUN is the response to MODIFY even though the two words do not appear on the same line.

ENTER '// ' CONTROL STATEMENT

// UIN UNIT-'F2,R2',TYPE-PRIMARY

ENTER '// ' CONTROL STATEMENT

// VOL PACK-2222

ENTER '// ' CONTROL STATEMENT

// VOL PACK-PAYROL,ID-010270

ENTER '// ' CONTROL STATEMENT

// END

Message printed by Disk Initialization program.

Control statement supplied by operator.

Sequence repeats until operator enters END statement.

#### Explanation:

- Disk Initialization program is loaded from the fixed disk on drive 1 (UNIT-F1 in OCL sequence).
- The two disks on drive 2 are being initialized (UNIT-'F2,R2' in UIN statement).
- The fixed disk (F2) will be given the name 2222 (PACK-2222 in first VOL statement).
- The removable disk (R2) will be given the name PAYROL (PACK-PAYROL in second VOL statement). Additional identifying information, 010270, will be written on the removable disk (ID-010270).

:

## MESSAGES FOR DISK INITIALIZATION

Message	Meaning
<p>INITIALIZATION ON XX COMPLETE</p>	<p>This message is printed when initialization of a disk is complete. XX indicates the unit (R1, R2, F1, or F2) on which the initialization is complete.</p>
<p>INITIALIZATION ON XX TERMINATED</p>	<p>This message is printed when initialization of a disk must be terminated for one of the following reasons:</p> <ol style="list-style-type: none"> <li>1. Cylinder zero is defective.</li> <li>2. More than six tracks are defective.</li> <li>3. Possible disk hardware error exists.</li> <li>4. The program attempted to initialize the disk ten times without success.</li> </ol> <p>After this message is printed, halt A13 will occur. XX indicates the unit (R1, R2, F1, or F2) on which the initialization is terminated.</p>
<p>**ALTERNATE TRACKS ASSIGNED**</p>	<p>These two messages are printed when a primary track is defective and an alternate track is assigned to it.</p>
<p>PRIMARY TRACK XXX ALTERNATE TRACK XXX</p>	<p>XXX indicates the tracks involved.</p>
<p>UNRECOVERABLE ERROR; RE-INITIALIZING PACK</p>	<p>This message is printed when the Disk Initialization program determines that the disk has not been initialized properly. The program will again attempt to initialize the disk correctly with ERASE-YES forced. The maximum number of times that the program will attempt to initialize a disk is ten. After that number of times, halt A13 occurs.</p>



## ALTERNATE TRACK ASSIGNMENT PROGRAM

The Alternate Track Assignment program (\$ALT) assigns alternate tracks to disk tracks that become defective after they are initialized. An alternate track is a track that can be assigned to replace another track. When the program assigns an alternate, it transfers the contents of the defective track to the alternate. Every disk

has six alternate tracks. An alternate track can replace any track except tracks 0 and 1 or another alternate track.

The program has three uses. The control statements you must supply depend on the program use.

Program Use	Situation
<p><b>Conditional assignment.</b> Program tests the condition of a track and assigns an alternate to it if it is defective. (This is the normal use.)</p> <p><b>Unconditional assignment.</b> ① Program assumes the track is defective and assigns an alternate to it without testing its condition.</p> <p><b>Cancel prior assignment.</b> ① Program cancels alternate-track assignment to free the alternate for use with another track.</p>	<p>Anytime a disk track causes reading or writing errors during a job, the system prints a message requesting that you run the Alternate Track Assignment program. You would normally use the program to do conditional assignment.</p> <p>You have used the Alternate Track Assignment program to do conditional assignment. The test on the track indicated that the track was not defective (an alternate, therefore, was not assigned). But the track still causes reading or writing errors, and you want to assign an alternate to it.</p> <p>A defective track was found, but all alternates are in use. You want to free an alternate so you can recover the data from the defective track. Before freeing the alternate, however, you would normally copy (to another disk) the file or library entry that uses the alternate. This saves the data that is already on the alternate. Run the File and Volume Label Display Program to determine which tracks are assigned alternates.</p>

① Conditional assignment is forced each time after an unconditional request.

## CONTROL STATEMENT SUMMARY FOR \$ALT

– Use –

– Control Statements – ①

Conditional Assignment

```
// ALT PACK-name,UNIT-code,VERIFY-number ②  
// END
```

Unconditional Assignment

```
// ALT PACK-name,UNIT-code,ASSIGN- { track }  
                                  { 'tracks' } ,VERIFY-number ③  
// END
```

Cancel Prior Assignment

```
// ALT PACK-name,UNIT-code,UNASSIGN- { track }  
                                       { 'tracks' } ,VERIFY-number ③  
// END
```

- ① For each use, the program requires the statements in the order they are listed: ALT, END.
- ② Optional parameter.
- ③ Optional parameter; applies to the automatic conditional assignment.



## PARAMETER SUMMARY

### ALT (Alternate) Statement

PACK-name

Name of the disk.

UNIT-code

Location of the disk. Possible codes are R1, F1, R2, F2.

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 once.

ASSIGN-track

Assign an alternate (unconditionally) to one track.

ASSIGN-'track,track,...'

Assign one alternate unconditionally to each track (maximum is six).

UNASSIGN-track

Cancel one alternate-track assignment. ①

UNASSIGN-'track,track,...'

Cancel two or more alternate-track assignments (maximum is six). ①

Use track numbers (8-405) to identify tracks. Tracks 0-7 are used by the system and cannot be assigned alternates.

Use track numbers (8-405) to which alternates are assigned.

- ① Before cancelling an assignment, the program tests the condition of the track to which the alternate is assigned. The assignment is cancelled if the test indicates that the track is not defective. If the test indicates that the track is defective, the program does not cancel the assignment unless the operator tells it to do so.

## 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.

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 as follows:

Code	Location
R1	Removable disk on drive 1.
F1	Fixed disk on drive 1.
R2	Removable disk on drive 2.
F2	Fixed disk on drive 2.

### 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 1-255. If you omit the parameter, the program does surface analysis once.

#### 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. It is the normal use of the Alternate Track Assignment program.

Situation: Conditional assignment applies to tracks that cause reading or writing errors during a job. Anytime 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.

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, fixed and removable, 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 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 once. 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.

There are six alternate tracks. The program will not do conditional assignment if all six are already in use.

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 sectors 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 45...
      B    A
      6    5
```

Appendix J 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.

### **ASSIGN Parameter**

The ASSIGN parameter (ASSIGN-track) applies to unconditional assignment. It tells the program which tracks you want alternates assigned to.

You can assign alternates to any tracks except 0-7. Tracks 0-7 are for system use only.

The form of the ASSIGN parameter depends on the number of tracks you want to specify. For one track, use ASSIGN-track; for two tracks, use ASSIGN-'track,track'; and so on. You can specify up to six tracks.

Use the track numbers (8-405) to identify the tracks. For example, the parameter ASSIGN-'50,301,353' causes the program to assign alternate tracks to tracks 50, 301, and 353.

### **Unconditional Assignment**

Unconditional assignment applies to tracks that occasionally cause read or write errors. Such tracks might not cause errors when tested by the Alternate Track Assignment program during conditional assignment. If they don't, the program will not assign alternate tracks to them. If you still want to assign alternates to these tracks, use unconditional assignment. In doing unconditional assignment, the program assigns alternates without first testing the condition of the tracks suspected of being defective.

### **UNASSIGN Parameter**

The UNASSIGN parameter (UNASSIGN-track) applies to cancelling alternate track assignments. It identifies tracks for which you want the program to cancel assignments.

You can cancel up to six assignments. The form of the UNASSIGN parameter depends on the number of assignments you want to cancel. For one assignment, use UNASSIGN-track; for two assignments, use UNASSIGN-'track,track'; and so on.

Use the track numbers (8-405) to identify the tracks. For example, the parameter UNASSIGN-'50,301,352' causes the program to cancel alternate-track assignments for tracks 50, 301, and 352.

### **Cancel Prior Assignment**

Cancelling an alternate-track assignment consists of transferring the data from an alternate track back to the original track (the track to which the alternate is assigned), therefore freeing the alternate from being the substitute for the original track.

Before transferring data back to the original track, the Alternate Track Assignment program tests the condition of the original track. If the test indicates that the track is defective, the program stops. Through the restart procedure you choose, you can tell the program to do one of three things:

1. Leave the assignment as it is. If there are other tracks for which you are cancelling assignments, the program continues with those. Otherwise it ends.
2. Cancel the assignment and transfer the data back to the original track regardless of the condition of the original track.
3. Test the track again.

Cancelling assignments is not often done. It applies to cases where a defective track is found, but all six alternates are in use. To recover the data from the defective track, you might want to cancel an alternate-track assignment to free the alternate track. Normally this involves copying, to another disk, a file or library entry that uses an alternate track, then freeing the alternate for use with the defective track you found. Run the File and Volume Label Display Program to determine what tracks are assigned alternates.

## OCL CONSIDERATIONS

LOAD Sequence		
<i>Keywords</i>	<i>Responses</i>	<i>Considerations</i>
READY	LOAD	-----
LOAD NAME	\$ALT	Name of Alternate Track Assignment program.
UNIT	R1, R2, F1, or F2	Location of disk containing Alternate Track Assignment program.
MODIFY	RUN	-----
↑ Only the key-words listed here are required. You can bypass the rest.	↑ You end every response by pressing PROG START.	

BUILD Sequence		
<i>Keywords</i>	<i>Responses</i>	<i>Considerations</i>
READY	BUILD	-----
BUILD NAME	procedure name	Name by which procedure will be identified in source library.
UNIT	R1, R2, F1, or F2	Location of disk containing source library.
LOAD NAME	\$ALT	Name of Alternate Track Assignment program.
UNIT	R1, R2, F1, or F2	Location of disk containing Alternate Track Assignment program.
MODIFY	<ul style="list-style-type: none"> <li>• INCLUDE utility control statements</li> <li>RUN</li> </ul>	Response when including control statements in procedure.
	<ul style="list-style-type: none"> <li>• RUN</li> </ul>	Response when not including control statements in procedure.
↑ Only the key-words listed here are required. You can bypass the rest.	↑ You end every response by pressing PROG START.	

## EXAMPLE

### Conditional Assignment

#### Situation

Assume that during a job the system printed a message telling the operator it found a defective track on the removable disk on drive 1. (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.

#### Statements

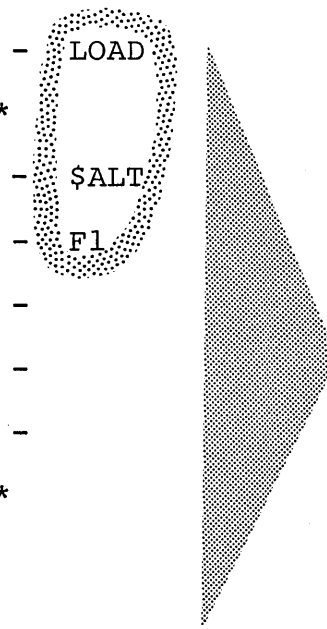
```

READY                                - LOAD
*****
010  LOAD  NAME                       - $ALT
011                UNIT               - F1
020  DATE   (XX/XX/XX)                -
030  SWITCH (00000000)                -
040  FILE   NAME                      -
*****

MODIFY
RUN

ENTER '//' CONTROL STATEMENT
// ALT PACK-BILLING,UNIT-R1
ENTER '//' CONTROL STATEMENT
// END

```



**OCL LOAD Sequence**

Circled areas are operator responses.

Keywords for which no responses are shown are the ones bypassed. If you press ENTER— after responding to UNIT, the DATE, SWITCH, and FILE NAME keywords are not prompted.

RUN is the response to MODIFY even though the two words do not appear on the same line.

Message printed by Alternate Track Assignment program.

Control statement supplied by operator.

System reprompts. END statement terminates sequence.

#### Explanation

- Alternate Track Assignment program is loaded from the fixed disk on drive 1 (UNIT-F1 in OCL sequence).
- The name of the disk (BILLNG) and its location (removable disk on drive 1) 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 once when it tests the condition of the track.

## 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	This message is printed when it is determined that a primary track is not defective.
PRIMARY TRACK STILL DEFECTIVE	This message is printed when the Alternate Track Assignment program determines that the track is still defective.
DATA TRANSFERRED BACK TO PRIMARY TRACK	This message is printed when the data is transferred back to the primary track.
**SECTOR WITH DATA ERROR**	This message is printed when the Alternate Track Assignment program found an error when transferring data. The sector that has the error is printed out.
PRIMARY TRACK xxx ALTERNATE TRACK yyy, UNIT-zz	This message is printed after ALTERNATE TRACK ASSIGNED and DATA TRANSFERRED BACK TO PRIMARY TRACK. xxx is the primary track number, yyy is the alternate track number, and zz is the unit involved.

The Alternate Track Rebuild program (\$BUILD) enables you to correct data that could not be transferred correctly to an alternate track. Many 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 in the listing

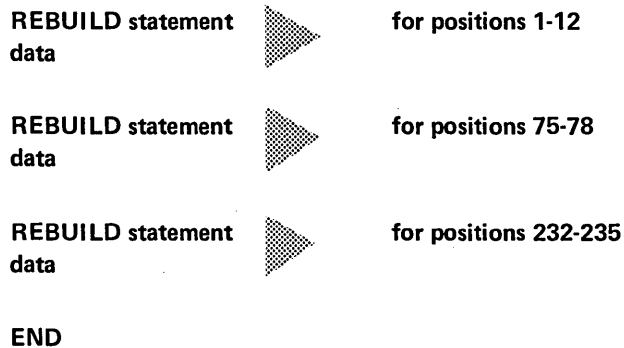
printed by the Alternate Track Assignment program when it assigned the alternate track. The listing tells you the name of the disk and numbers of the track and sectors suspected of containing incorrect data. It also includes the data from these sectors, which you can use to locate incorrect data.

**CONTROL STATEMENT SUMMARY FOR \$BUILD**

```
// REBUILD PACK-name,UNIT-code,TRACK-location,LENGTH-number,DISP-position ①  
  
Substitute data  
  
// END
```

① At least one REBUILD statement is needed for every sector you correct. If the characters you replace in a sector occupy consecutive positions, you need only one REBUILD statement for that sector. Otherwise, you need one statement for every group of characters that do not occupy consecutive positions. For example, to replace characters 1-12, 75-78, and 232-235 in a sector, you would need three REBUILD statements.

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



## PARAMETER AND SUBSTITUTE DATA SUMMARY

### REBUILD Statement

**PACK-name**

Name of the disk.

**UNIT-code**

Location of the disk. Possible codes are R1, F1, R2, F2.

**TRACK-location**

Number of track and sector containing incorrect data. Number is printed by Alternate Track Assignment program. Track number must be three digits. Sector number must be two digits. For example:

TRACK-01109 means track 11, sector 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 sector. Position can be 1-255.

### Substitute Data

Key each character in hexadecimal form. Follow every second character, except the last, with a comma. EXAMPLE: The numbers 123456 would be keyed as F1F2, F3F4, F5F6.



## 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 ensure 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 as follows:

Code	Location
R1	Removable disk on drive 1.
F1	Fixed disk on drive 1.
R2	Removable disk on drive 2.
F2	Fixed disk on drive 2.

### TRACK Parameter

The TRACK parameter (TRACK-location) identifies the track and sector that contains 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.

Use the track and sector numbers in the TRACK parameter. The possible track numbers are 008-405. Always use three digits. The possible sector numbers are 00-23. Always use two digits. The track number must precede the sector number. For example, the parameter TRACK-11019 means track 110, sector 19.

Track and sector numbers are printed by the Alternate Track Assignment program when it prints data from sectors that contain incorrect data.

### LENGTH Parameter

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

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

1. Use one REBUILD statement to replace characters 10-25 (LENGTH-16).
2. 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 sector. The position of the first character in the sector is 1; the position of the second character is 2; and so on. The maximum position 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 key the substitute characters that apply to that statement. The characters must be in hexadecimal form. Appendix J shows the hexadecimal forms of the characters in the standard 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.

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

## OCL CONSIDERATIONS

<b>LOAD Sequence</b>		
<i>Keywords</i>	<i>Responses</i>	<i>Considerations</i>
READY	LOAD	-----
LOAD NAME	\$BUILD	Name of Alternate Track Rebuild program.
UNIT	R1, R2, F1, or F2	Location of disk containing Alternate Track Rebuild program.
MODIFY	RUN	-----
↑ Only the key-words listed here are required. You can bypass the rest.	↑ You end every response by pressing PROG START.	

<b>BUILD Sequence</b>		
<i>Keywords</i>	<i>Responses</i>	<i>Considerations</i>
READY	BUILD	-----
BUILD NAME	procedure name	Name by which procedure will be identified in source library.
UNIT	R1, R2, F1 or F2	Location of disk containing source library.
LOAD NAME	\$BUILD	Name of Alternate Track Rebuild program.
UNIT	R1, R2, F1 or F2	Location of disk containing Alternate Track Rebuild program.
MODIFY	RUN*	Response when not including control statements in procedure.
↑ Only the key-words listed here are required. You can bypass the rest.	↑ You end every response by pressing PROG START.	

\*\$BUILD does not allow utility control statements in the procedure.

## EXAMPLE

### Correcting Characters on an Alternate Track

#### Situation

Assume that the Alternate Track Assignment program printed the following information:

PACK-R1

TRACK AND SECTOR BAD-05020

ABCDEF GH1 34567890... (Assume the entire contents of the sector  
was printed.)  
B A  
6 5

It means that errors were detected in sector 20 of track 50 on the removable disk on drive 1. (Assume the name of the disk is BILLNG.)

In checking the characters printed by the program, you found that the seventh and eleventh characters in the sector are incorrect and you want the operator to run the Alternate Track Rebuild program to correct them.

**Statements**

```

READY          - LOAD
*****
010  LOAD      NAME      - $BUILD
011                UNIT  - F1
020  DATE      (XX/XX/XX) -
030  SWITCH    (00000000) -
040  FILE      NAME      -
*****
MODIFY
RUN
    
```

**OCL LOAD Sequence**

Circled areas are operator responses.

Keywords for which no responses are shown are the ones bypassed. If you press ENTER— after responding to UNIT, the DATE, SWITCH, and FILE NAME keywords are not prompted.

RUN is the response to MODIFY even though the two words do not appear on the same line.

```

ENTER '//' CONTROL STATEMENT
// REBUILD PACK-BILLING,UNIT-R1,TRACK-05020,LENGTH-6,DISP-7
ENTER HEX DATA STATEMENT
C6C7,C8F1,F2F3
ENTER '//' CONTROL STATEMENT
// END
    
```

Message printed by Alternate Track Rebuild program.

Message printed by Alternate Track Rebuild program.

Message printed by Alternate Track Rebuild program.

Control statements and substitute data supplied by the operator

**Explanation**

- Alternate Track Rebuild program is loaded from the fixed disk on drive 1 (UNIT-F1 in OCL sequence).
- The name of the removable disk (BILLNG) and its location (drive 1) are indicated in the PACK and UNIT parameters in the REBUILD statement.
- The sector containing the incorrect characters is sector 20 of the alternate track assigned to track 50 (TRACK-05020). The seventh character in the sector is the first character being replaced (DISP-7).
- The seventh through twelfth characters in sector 20 are being replaced (LENGTH-6). We included the twelfth character because the number of characters being replaced must be a multiple of 2. By also replacing the characters between the incorrect ones, we needed only one REBUILD statement.
- The substitute characters follow the REBUILD statement. They are F (C6), G (C7), H (C8), 1 (F1), 2 (F2), and 3 (F3).

## FILE AND VOLUME LABEL DISPLAY PROGRAM

The File and Volume Label Display program (\$LABEL) has two uses:

1. Print the entire Volume Table of Contents (VTOC) from a disk.
2. Print 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:

1. 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.
2. You want to find out what disk areas are available for libraries or new files.
3. 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 FOR \$LABEL

– Uses –

– Control Statements – ①

Print entire VTOC

```
// DISPLAY UNIT-code, LABEL-VTOC
// END
```

Print only file information from VTOC

```
// DISPLAY UNIT-code, LABEL- {filename } ②
                             {'filenames'}
// END
```

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

② More than one DISPLAY statement may be used before the END statement. However, the total number of filenames on all the DISPLAY statements cannot exceed 20, where VTOC is considered as one name.

## PARAMETER SUMMARY

### DISPLAY Statement

UNIT-code

Location of the disk. Possible codes are R1, F1, R2, F2.

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. You may list as many filenames as the statement will hold. The control statement length is restricted to 96 characters. Maximum is 20 filenames on all DISPLAY statements.

## 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 as follows:

<i>Code</i>	<i>Location</i>
R1	Removable disk on drive 1.
F1	Fixed disk on drive 1.
R2	Removable disk on drive 2.
F2	Fixed disk on drive 2.

### LABEL Parameter

The LABEL parameter indicates the information you wanted 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. Every disk, fixed and removable, contains a VTOC.

#### 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.

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.

**Meaning of VTOC Information**

– Heading –

– Meaning –

PACK-name

Name of the disk.

ID-characters

Additional disk identification (if any).

NUMBER OF ALTERNATE TRACKS AVAILABLE-number

Number of alternate tracks available for assignment.

TRACKS WITH ALTERNATE ASSIGNED

Tracks that have an alternate assigned to them.

DEFECTIVE ALTERNATE TRACKS

Numbers of the alternate tracks that are defective.

DEVICE CAPACITY-number

Disk drive capacity (number of tracks).

LIBRARY EXTENT

Boundary of libraries on the disk. (If the disk contains no libraries, these headings are not printed.)

START

Track on which library begins.

END

Track on which library ends.



If disk contains both source and object library START refers to beginning of source library and END refers to end of object library.

EXTENDED END

Object library 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.

TRACKS

Number of tracks available.

PACK-name

Name of the disk.

UNIT-code

Location of disk containing VTOC information

DATE-xx/xx/xx

Current system date.

FILE NAME

Name that identifies file in VTOC.

FILE DATE

Date given the file when file was placed on disk.

PACK-name  
UNIT-code (continued)

KEEP TYPE	File designation: P=permanent. T=temporary. S=scratch.
FILE TYPE	File type: I=Indexed.                   D=Direct. C=Consecutive.            B=BASIC.
REC LEN	Number of characters in each record in file.
KEY LEN	Indexed files only. Number of characters in each record key.
KEY LOC	Indexed files only. Position in record occupied by last character of record key.
NEXT AVAIL RECORD	Beginning location of next available record in file. Location is track, sector, and position within sector. EXAMPLE: 09918006=track 99, sector 18, position 6. ①
NEXT AVAIL KEY	Indexed files only. Beginning location of next available key in index portion of file. Location is track, sector, and position within sector. EXAMPLE: 09010006=track 90, sector 10, position 6. ②
INDEX START END	Indexed files only. Tracks on which index starts (START) and ends (END).
DATA START END	Disk area reserved for the file. START is the first track of the area. END is the last track. For indexed files, this refers to the data portion of the file.
VOL SEQ	VOL SEQ applies to multi-volume files only. It indicates the order of this disk as it relates to the other disks containing the remaining portions of the file.

① If the first byte of the next available record occurs in the next track after the end track of DATA START END 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, then the 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 as many filenames as the statement will hold. The statement length, however, is restricted to 96 characters. Maximum is 20 filenames on all DISPLAY statements.

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 in the preceding chart, *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

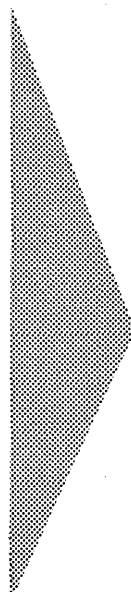
<b>LOAD Sequence</b>		
<i>Keywords</i>	<i>Responses</i>	<i>Considerations</i>
READY	LOAD	-----
LOAD NAME	\$LABEL	Name of File and Volume Label Display program.
UNIT	R1, R2, F1, or F2	Location of disk containing File and Volume Label Display program.
MODIFY	RUN	-----
↑ Only the key-words listed here are required. You can bypass the rest.	↑ You end every response by pressing PROG START.	

<b>BUILD Sequence</b>		
<i>Keywords</i>	<i>Responses</i>	<i>Considerations</i>
READY	BUILD	-----
BUILD NAME	procedure name	Name by which procedure will be identified in source library.
UNIT	R1, R2, F1, or F2	Location of disk containing source library.
LOAD NAME	\$LABEL	Name of File and Volume Label Display program.
UNIT	R1, R2, F1, or F2	Location of disk containing File and Volume Label Display program.
MODIFY	<ul style="list-style-type: none"> <li>• INCLUDE utility control statements</li> <li>RUN</li> <li>• RUN</li> </ul>	Response when including control statements in procedure.  Response when not including control statements in procedure.
↑ Only the key-words listed here are required. You can bypass the rest.	↑ You end every response by pressing PROG START.	

**EXAMPLE**

**Printing VTOC Information for Two Files**

```
READY - LOAD
*****
010 LOAD NAME - $LABEL
011 UNIT - F1
020 DATE (XX/XX/XX) -
030 SWITCH (00000000) -
040 FILE NAME -
*****
MODIFY
RUN
```



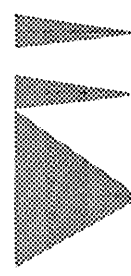
**OCL LOAD Sequence**

Circled areas are operator responses.

Keywords for which no responses are shown are the ones bypassed. If you press ENTER- after responding to UNIT, the DATE, SWITCH, and FILENAME keywords are not prompted.

RUN is the response to MODIFY even though the two words are not on the same line.

```
ENTER '// ' CONTROL STATEMENT
// DISPLAY UNIT-R1,LABEL-'BILLING,INV01'
ENTER '// ' CONTROL STATEMENT
// DISPLAY UNIT-F2,LABEL-VTOC
ENTER '// ' CONTROL STATEMENT
// END
```



Message printed by File and Volume Label Display program.

Control statement supplied by operator.

Sequence repeats until operator enters END statement.

**Explanation:**

- The File and Volume Label Display program is loaded from the fixed disk on drive 1 (UNIT-F1 in OCL sequence).
- The files for which information is printed are named BILLING and INVO1 (LABEL-'BILLING,INV01' in first DISPLAY statement). They are located on the removable disk on drive 1 (UNIT-R1).
- Information from the entire VTOC on F2 is printed.

The File Delete program (\$DELET) has three uses:

- Remove all files from a disk.
- Remove only the files you name.
- Scratch file references in the Volume Table of Contents (VTOC).

Deleting files frees the space they occupy for use by new files.

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 program depend on the program use.

# CONTROL STATEMENT SUMMARY FOR \$DELET

— USE —

— CONTROL STATEMENTS <sup>①</sup> —

Scratch all files  
in the VTOC.

```
// SCRATCH PACK-name, UNIT-code, LABEL-VTOC
// END
```

Scratch only the  
files named in  
the VTOC

```
// SCRATCH PACK-name, UNIT-code, LABEL-filename, DATE-date ②
// END
```

```
// SCRATCH PACK-name, UNIT-code, LABEL- { filename }
// END { 'filenames' }
```

Remove all files  
from the disk.

```
// REMOVE PACK-name, UNIT-code, LABEL-VTOC,
DATA- { NO }
// END { YES }
```

Remove only the  
files named from  
the disk.

```
// REMOVE PACK-name, UNIT-code, LABEL-filename,
DATE-date, DATA- { NO } ②
// END { YES }
```

```
// REMOVE PACK-name, UNIT-CODE, LABEL- { filename }
DATA- { NO }
// END { 'filenames' }
```

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

The SCRATCH statement does not erase files from the disk. It changes their designation to scratch (S) in the Volume Table of Contents (VTOC). By doing this, the program makes the areas that contain the files available for other files. A halt will occur if an attempt is made to create a new multi-volume file that will have the same label on disk as an existing single volume file, or if an attempt is made to create a single volume file bearing the same label as an existing multi-volume file. The halt will occur even though the retain on the existing file is scratch. If a REMOVE statement is used, files are erased from the disk. No file is physically scratched or removed from the VTOC until end of job has occurred.

- 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. At least one SCRATCH or REMOVE statement is required by the program. When deleting files, you can list as many filenames as the statement will hold. The statement length, however, cannot exceed 96 characters. If you want to delete more files than you can specify in one SCRATCH or REMOVE statement, use additional statements. The END statement must follow the last SCRATCH or REMOVE statement.

## PARAMETER SUMMARY

### Scratch Statement

**PACK-name**

Name of the disk.

**UNIT-code**

Location of the disk. Possible codes are R1, F1, R2, F2.

**LABEL-VTOC**

Scratch all files from VTOC.

**LABEL-filename**

Scratch only file named from VTOC.

**LABEL-'filename,filename,...'**

Scratch only the files named from VTOC. (You may list as many filenames as you want.)

Use names that identify files in VTOC. These are the names that you gave the files when you placed them on disk.

**DATE-date**

Date of the file being deleted. If two more more files have the same name you list in the LABEL parameter, they will all be deleted unless you use a DATE parameter to indicate a particular file.

Date must be a six-digit number. EXAMPLE: DATE-062070 means June 20, 1970.

## Remove Statement

**PACK-name**

Name of the disk.

**UNIT-code**

Location of the disk. Possible codes are R1, F1, R2, F2.

**LABEL-VTOC**

Delete all files from the disk.

**LABEL-filename**

Delete only the file named from the disk.

**LABEL-'filename,filename,...'**

Delete only the files named. (You may list as many filenames as you want.)

Use names that identify files in VTOC. These are the names that you gave the files when you placed them on disk.

**DATE-date**

Date of the file being deleted. If two more files have the same name you list in the LABEL parameter, they will all be deleted unless you use a DATE parameter to indicate a particular file.

Date must be a six-digit number. EXAMPLE: DATE-062070 means June 20, 1970.

**DATA**

{ NO }  
{ YES }

Removes the data for the referenced files from the disk.

## PARAMETER DESCRIPTIONS

ditional REMOVE or scratch statements may be used for additional filenames. The maximum number of files that can be deleted in one run is 52.)

### | 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 disk by the Disk Initialization program.

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 as follows:

Code	Location
R1	Removable disk on drive 1.
F1	Fixed disk on drive 1.
R2	Removable disk on drive 2.
F2	Fixed disk on drive 2.

### | 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. Ad-

### Deleting Files

The File Delete program does not erase files from the disk unless DATA-YES is specified on a REMOVE statement. It changes their designation to scratch (S). By doing this, the program makes the areas that contain the files available for other files.

### | DATE Parameter

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 tell one file from another.

The date is a six-digit number: two digits for day, two for month, and two for year. Day, month, and year can be in one of two orders: (1) month, day, year and (2) day, month, year. For example 061870 and 180670 both mean June 18, 1970.

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

### | DATA Parameter (Remove Only)

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

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

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

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



## OCL CONSIDERATIONS

LOAD Sequence		
<i>Keywords</i>	<i>Responses</i>	<i>Considerations</i>
READY	LOAD	-----
LOAD NAME	\$DELET	Name of File Delete program.
UNIT	R1, R2, F1, or F2	Location of disk containing File Delete program.
MODIFY	RUN	-----
↑ Only the key-words listed here are required. You can bypass the rest.	↑ You end every response by pressing PROG START.	

BUILD Sequence		
<i>Keywords</i>	<i>Responses</i>	<i>Considerations</i>
READY	BUILD	-----
BUILD NAME	procedure name	Name by which procedure will be identified in source library.
UNIT	R1, R2, F1, or F2	Location of disk containing source library.
LOAD NAME	\$DELET	Name of File Delete program.
UNIT	R1, R2, F1, or F2	Location of disk containing File Delete program.
MODIFY	<ul style="list-style-type: none"> <li>• INCLUDE utility control statements R'JN</li> <li>• RUN</li> </ul>	Response when including control statements in procedure.  Response when not including control statements in procedure.
↑ Only the key-words listed here are required. You can bypass the rest.	↑ You end every response by pressing PROG START.	

## EXAMPLE

### Deleting One of Several Files Having the Same Name

#### Situation

Assume that three files on a removable disk have the same name: INV01. The dates of these files are 6/16/70, 8/18/70, and 11/15/70. You want to delete the 6/16/70 version.

#### Statements

READY

\*\*\*\*\*

010 LOAD NAME - \$DELET

011 UNIT - F1

020 DATE (XX/XX/XX) -

030 SWITCH (00000000) -

040 FILE NAME -

\*\*\*\*\*

MODIFY

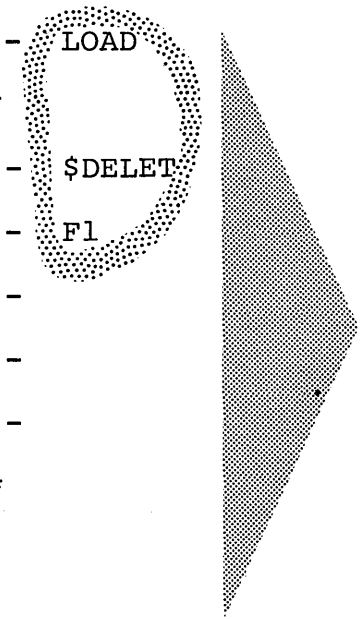
RUN

ENTER '//' CONTROL STATEMENT

// SCRATCH PACK-00001,LABEL-INV01,UNIT-R1,DATE-061670

ENTER '//' CONTROL STATEMENT

// END



#### OCL Load Sequence

Circled areas are operator responses.

Keywords for which no responses are shown are the ones bypassed. If you press ENTER— after responding to UNIT, the DATE, SWITCH, and FILE NAME keywords are not prompted.

RUN is the response to MODIFY even though the two words do not appear on the same line.

Message printed by File Delete program.

Control statement

supplied by operator.

Sequence repeats until operator enters END statement.

#### Explanation

- File Delete program is loaded from the fixed disk on drive 1 (UNIT-F1 in OCL sequence).
- Disk that contains the file being deleted is named 00001 (PACK-00001 in SCRATCH statement).
- Because two other files have the name INV01, the date (061670) is needed to complete the identification of the file you want to delete (LABEL-INV01 and DATE-061670).
- The removable disk containing the file to be deleted is on drive 1 (UNIT-R1).

## Removing One File

### Situation

You want to remove a file named INV02 from the pack mounted on R1.

### Statements

```

READY
*****
010  LOAD  NAME  -
011          UNIT  -
020  DATE  (XX/XX/XX) -
030  SWITCH (00000000) -
040  FILE  NAME  -
*****

```

MODIFY

RUN

LOAD  
\$DELET  
F1

### OCL Load Sequence

Circled areas are operator responses.

Keywords for which no responses are shown are the ones bypassed. If you press ENTER— after responding to UNIT, the DATE, SWITCH, and FILE NAME keywords are not prompted.

RUN is the response to MODIFY even though the two words do not appear on the same line.

ENTER '// ' CONTROL STATEMENT

Message printed by File Delete program.

// REMOVE PACK-00001 , LABEL-INV02 , UNIT-R1 , DATA-YES

Control statement supplied by operator.

'DATA REMOVED FOR FILE xxxxxx DATE 000000'

Printed by File Delete.

ENTER '// ' CONTROL STATEMENT  
// END

Sequence repeats until operator enters END statement.

### Explanation

- File Delete program is loaded from the fixed disk on drive 1 (UNIT-F1 in OCL sequence).
- Disk that contains the file being removed is named 00001 (PACK-00001 in REMOVE statement).
- The removable disk containing the file to be removed is on drive 1 (UNIT-R1).
- DATA-YES indicates that the file data as well as the file VTOC reference is to be removed.



The Disk Copy/Dump program (\$COPY) has three general uses. The control statements you must supply depend on the program use.

Program Uses	Common Reasons
<p>Copy entire contents of one disk to another.</p> <p>Copy a data file from one disk to another, or from one area to another on same disk.</p> <p>Print all or part of a data file.</p>	<p>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.</p> <p>Any of the following:</p> <ul style="list-style-type: none"> <li>● Provide a reserve file in case something happens to the original file.</li> <li>● Move a file to a larger disk area.</li> <li>● Reorganize the data portion of an indexed file. (Data in the copy of the file is reorganized; the original file is unchanged.)</li> <li>● Delete records from a file. (Records are omitted from the copy of the file; the original file remains unchanged.)</li> </ul> <p>Provide a printed copy of the records in a file, perhaps for use in checking the records for errors.</p>
<p>Your responses to file keywords in the OCL sequence used to load the program describe the disk file being copied or printed. If you are copying the file to disk, the file being created must also be described in the OCL sequence.</p>	

CONTROL STATEMENT SUMMARY FOR \$COPY

- USES ① -

- Control Statements ② -

Copy an Entire Disk R1 R1  
 // COPYPACK FROM-code,TO-code  
 // END

Copy a Data File  
 //COPYFILE OUTPUT-DISK,DELETE-'position,character', REORG- { NO } ④ WORK- { NO } ⑤  
 or OUTPTX or OMIT { YES }  
 // END

Copy and Print a Data File  
 //COPYFILE OUTPUT-BOTH,DELETE-'position,character', REORG-YES, ④ WORK- { NO } ⑤  
 or OUTPTX or OMIT { YES }  
 // END

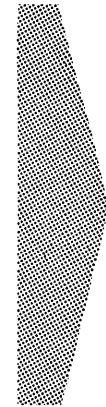
Copy a Data File, But Print Only a Part of the File  
 // COPYFILE OUTPUT-BOTH,DELETE-'position,character', REORG-YES, ④ WORK- { NO } ⑤  
 or OUTPTX or OMIT { YES }  
 // SELECT KEY, FROM-'key' ④  
 -or-  
 // SELECT KEY, FROM-'key', TO-'key' ④  
 -or-  
 // SELECT PKY, FROM-'key' ⑦  
 -or-  
 // SELECT PKY, FROM-'key', TO-'key' ⑦  
 -or-  
 // SELECT RECORD, FROM-number  
 -or-  
 // SELECT RECORD, FROM-number, TO-number  
 // END

One of these. ⑥

Print an Entire Data File  
 // COPYFILE OUTPUT-PRINT  
 or OUTPTX  
 // END

Print Only a  
Part of a  
Data File

```
// COPYFILE OUTPUT-PRINT  
      or  
      OUTPTX  
// SELECT KEY, FROM-'key' ④  
      -or-  
// SELECT KEY, FROM-'key', TO-'key' ④  
      -or-  
// SELECT PKY, FROM-'key' ⑦  
      -or-  
// SELECT PKY, FROM-'key', TO-'key' ⑦  
      -or-  
// SELECT RECORD, FROM-number  
      -or-  
// SELECT RECORD, FROM-number, TO-number  
  
// END
```



One  
of  
these. ⑥

- ① The program uses include the possible combinations 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; and COPYFILE,SELECT,END.
- ③ Needed only if you want to delete a certain type of record.
- ④ Applies only to indexed files.
- ⑤ Applies only if you are copying the file from one removable disk to another using the same disk drive (drive 1).
- ⑥ Identifies the portion you want to print.
- ⑦ Indexed files with packed keys.

## PARAMETER SUMMARY

### COPYPACK Statement

FROM-code

Location of disk to be copied. Possible codes are R1, F1, R2, F2.

TO-code

Location of disk to contain the copy. Possible codes are R1, F1, R2, F2.

### COPYFILE Statement

OUTPUT-DISK

-or-

OUTPTX-DISK

Copy the file from one disk to another, or from one area to another on the same disk. ①

OUTPUT-PRINT

-or-

OUTPTX-PRINT

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

OUTPUT-BOTH

-or-

OUTPTX-BOTH

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

DELETE-'position, character'

-or-

OMIT-'position, character'

These parameters are optional. All records with the specified character in the specified record position are deleted. DELETE causes deleted records to be printed. Character can be any of the System/3 characters except blank, comma, or apostrophe. Position can be any position in the record (the first position is 1, second 2, and so on). The maximum position is 999.

REORG-NO

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-NO is assumed if you omit the REORG keyword.

REORG-YES

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. When OUTPUT-BOTH is used, REORG-YES is required.



**COPYFILE Statement (continued)**

**WORK-NO**

May be used in all cases except when copying a file from one removable disk to another on drive 1. It means: do not use a work area on the fixed disk on drive 1. WORK-NO is assumed if you omit the WORK keyword.

**WORK-YES**

Required for copying a file from one removable disk on drive 1 to another removable disk on that drive. It means: use a work area on the fixed disk on drive 1. WORK-NO is assumed if you omit the WORK keyword.

**SELECT Statement**

**KEY, FROM-'key'**  
-or-  
**PKY, FROM-'key'**

Indexed files only. Print only the part of the file from the record key that is specified in the FROM parameter to the end of the file.

**KEY, FROM-'key', TO-'key'**  
-or-  
**PKY, FROM-'key', TO-'key'**

Indexed files only. Print 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 only one record, make the FROM and TO record keys the same.

**RECORD, FROM-number**

Print 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 only the part of the file between the relative record numbers indicated by the parameters (including the records indicated by the parameter). To print only one record, make the FROM and TO record numbers the same.

- ① In his responses to OCL keywords (FILE NAME, etc.), the operator indicates which file is to be copied or printed. For files being copied, his responses also indicate whether the file is being copied from one disk to another or from one location to another on the same disk.

## PARAMETER DESCRIPTIONS

### | FROM and TO Parameters (COPYPACK)

The COPYPACK statement is used to copy the contents of one disk to another. It has two parameters: FROM and TO. They tell the program the locations of the two disks on the disk units.

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.

Codes for the possible locations are as follows:

Code	Location
R1	Removable disk on drive 1.
F1	Fixed disk on drive 1.
R2	Removable disk on drive 2.
F2	Fixed disk on drive 2.

### Copying Entire Disk

When copying a disk, the Disk 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 have libraries, temporary data files, or permanent data files.

The program can copy the contents of one removable disk to another using one disk drive. The drive, however, must be drive 1. To do this, the program uses available space on the fixed disk on drive 1. It fills the available space with information from the disk you are copying. Then it prints a message telling the operator to mount the other removable disk (the one to contain the copy) on drive 1. After transferring the information from the fixed disk to the removable disk, the program prints another message telling the operator to remount the disk you are copying. The program repeats this procedure until all information has been transferred.

Until the contents of the disk are completely copied on the new disk, three addressing 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 pack is unusable. You can restart the copying process by reloading the copy program or you can restore the disk by reinitializing.

After a successful copy the copy program prints a message:

COPYPACK IS COMPLETE

### | OUTPUT Parameter (COPYFILE)

The OUTPUT parameter is used when copying and printing data files. It indicates whether you want the program to copy, print, or copy and print a file.

The parameter OUTPUT-DISK means to copy the file; OUTPUT-PRINT means to print the file; and OUTPUT-BOTH means to copy and print the file.

OUTPTX can be used instead of OUTPUT to display the printed output with its hexadecimal values.

### Copying Files

The Disk Copy/Dump program can copy a file from one disk to another or from one area to another on the same disk.

Your responses to the OCL keywords prompted for the Disk Copy/Dump program indicate (1) the name and location of the file being copied and (2) the name and location of the copy being created. See *OCL Considerations* in this section.

The program can copy a file from one removable disk to another using one disk drive. The drive, however, must be drive 1. (See *WORK Parameter* in this section for more information.)

In copying a file, the program can omit records. (See *DELETE Parameter* in this section for more information.)

In copying an indexed file, the program can reorganize records in the data portion such that they are in the same order as their keys are listed in the index. (See *REORG Parameter* in this section for more information.)

## Printing Files

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

If you use **SELECT** or **REORG**, 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. If **OUTPUT-** is specified, only printable characters are printed. If **OUTPTX-** is specified, all characters are printed with their 2-digit hexadecimal value. Appendix J lists the hexadecimal values for characters in the standard character set.

The following is an example of the way the program prints a 20-character record when **OUTPUT-** is specified.

```
ABCDE GHI J 12345
```

If **OUTPTX-** is specified, the same record would be printed:

```
ABCDE GHI J 12345  
CCCCBCCDFFFFF44444  
1 234 567891 1234500000
```

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

```
(number) RECORDS PRINTED
```

## DELETE Parameter (COPYFILE)

In copying a data file, the Disk Copy/Dump program can omit records of one type. The **DELETE** parameter identifies the type of records. Use of the **DELETE** parameter is optional. If you do not use it, no records are deleted.

The form of the parameter is **DELETE-'position, character'**. Character is the character, except apostrophes, blanks, and commas, that identifies the records. Position is the position of the character in the records (maximum 999). For example, with the parameter **DELETE-'100, X'** all records with an X 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 that are printed. 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, such 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 to reorganize. **REORG-NO** means not to reorganize. **REORG-NO** is assumed if you omit the keyword.

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 any time you are both copying and printing an indexed file (**OUTPUT-BOTH**).

## WORK Parameter (COPYFILE)

The **WORK** parameter applies to copying a data file from one removable disk to another using the same disk drive (drive 1). It tells the program whether or not to use a work area on the fixed disk on drive 1.

The parameter **WORK-YES** means to use a work area. **WORK-NO** means not to use a work area.

## Work Area

If you have only one disk drive, a common use of the Disk Copy/Dump program might be to copy a file from one removable disk to another. To do this, the program must use a work area on the fixed disk. The output file must be a new file.

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 removable disk (the one to contain the copy) on drive 1. After transferring the records from the work area to the removable disk, the program prints another message telling the operator to remount the disk containing the file you are copying. The program repeats this procedure until all records have been transferred.

If you have two disk drives, you can also use the same drive to copy a file from one removable disk to another. The drive, however, must be drive 1.

You can copy a file from one area to another on the same disk. If you do, and the disk is a removable disk that you plan to mount on drive 1, use the WORK-NO parameter (WORK-NO is assumed if the WORK keyword is not used). This keeps the program from using a work area on the fixed disk when it transfers the file from one area to the other.

### **SELECT KEY and PKY Parameters (SELECT)**

The SELECT KEY and SELECT PKY parameters apply to printing part of an indexed file. The parameters are FROM and TO.

The FROM parameter (FROM-'key') gives the key of the first record to be printed. The TO parameter (TO-'key') gives the key of the last record to be printed. The record keys between those two in the file index identify the remaining records to be printed. If you want to print 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 printed.

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.

With the SELECT KEY parameter (but not PKY) you can use less characters in the FROM or TO parameter than are contained in the actual keys. If you do, the program ignores the remaining characters in the 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: 99999, A1000, A1119, A1275, A1900, A1995, and A2075. The parameters FROM-'A1' and TO-'A199' refer to record keys A1000 through A1995.

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

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 the key beginning with the character 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 printed.

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 printed. The TO parameter (TO-number) gives the number of the last record to be printed. Records between those two records in the file are also printed. If you want to print only one record, use the same record number in the FROM and TO parameters.

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

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.

## **COPYING MULTI-VOLUME FILES**

When copying multi-volume files 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 neither condition is satisfied a halt occurs.

### **Maintaining Proper Volume Sequence Numbers**

To maintain proper volume sequence numbers when copying a multi-volume file, you must either copy all the volumes of the file in one run or copy only one volume for each run of \$COPY. For example, if you copy a 3-volume file one volume at a time: volume 1 in the first run, volume 2 in the second run, and volume 3 in the third run; the volumes will retain their original sequence numbers in the output file. Or if you copy all the volumes (1, 2, and 3) in the same run, the volume sequence numbers in the new file will be same as in the original file. However, if you copy only volumes 2 and 3 in one run, their volume sequence numbers will be changed to 1 and 2 in the output file.

### **Maintaining Correct Relative Record Numbers**

To maintain correct relative record numbers when copying one volume of a multi-volume direct file, the size of the output volume must be the same as the size of the input volume. (If you want to increase the size of a file, you must copy the entire file.) If, for example, 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, output and input volume extents must be equal if you are copying only one volume of a multi-volume direct file.

*Note:* You can not use the copy program to copy a single volume file to a multi-volume file. End of extents will probably occur after the first volume of output. If the output file is a new file, the copy program will not create it as a multi-volume file.

### **Direct File Attributes**

If you copy a whole multi-volume direct file in one run, the output file will be given consecutive attributes in the Volume Table of Contents (VTOC). However, this does not affect file processing. A file with either consecutive or direct attributes can be accessed by a consecutive or direct access method. If only one volume is copied, the direct attribute will be maintained.

### **Copying Multi-Volume Index Files**

If you want to copy a multi-volume indexed file, REORG-YES must be given. Since an unordered multi-volume indexed load is not permitted, a REORG-NO will cause a halt if an out-of-sequence record is found. If you would prefer not to reorganize the file, each volume of the file must be copied as a single volume file. When copying each volume separately, it can be either ordered or unordered. When copying one volume of a multi-volume indexed file, either REORG-YES or REORG-NO may be specified. HIKEY parameter(s) of the output file must be the same as the highest key(s) of each input volume.

## OCL CONSIDERATIONS

LOAD Sequence for Copying an Entire Disk		
<i>Keywords</i>	<i>Responses</i>	<i>Considerations</i>
READY	LOAD	-----
LOAD NAME	\$COPY	Name of Disk Copy/Dump program.
UNIT	R1, R2, F1, or F2	Location of disk containing Disk Copy/Dump program.
MODIFY	RUN	-----
↑ Only the key-words listed here are required. You can bypass the rest.	↑ You end every response by pressing PROG START.	

BUILD Sequence for Copying an Entire Disk		
<i>Keywords</i>	<i>Responses</i>	<i>Considerations</i>
READY	BUILD	-----
BUILD NAME	procedure name	Name by which procedure will be identified in source library.
UNIT	R1, R2, F1, or F2	Location of disk containing source library.
LOAD NAME	\$COPY	Name of Disk Copy/Dump program.
UNIT	R1, R2, F1, or F2	Location of disk containing Disk Copy/Dump program.
MODIFY	<ul style="list-style-type: none"> <li>• INCLUDE utility control statements</li> <li>RUN</li> </ul>	Response when including control statements in procedure.
	<ul style="list-style-type: none"> <li>• RUN</li> </ul>	Response when not including control statements in procedure.
↑ Only the key-words listed here are required. You can bypass the rest.	↑ You end every response by pressing PROG START.	

### LOAD Sequence for Copying or Printing Files

<i>Keywords</i>	<i>Responses</i>	<i>Considerations</i>
READY	LOAD	-----
LOAD NAME	\$COPY	Name of Disk Copy/Dump program.
UNIT	R1, R2, F1, or F2	Location of disk containing Disk Copy/Dump program.
FILE NAME	COPYIN	Name Disk Copy/Dump program uses to refer to file to be copied (input file).
UNIT	R1, R2, F1, or F2	Location of disk containing file to be copied.
PACK	disk name	Name of disk containing file to be copied.
LABEL	file name	Name by which file to be copied is identified on disk.
FILE NAME	<ul style="list-style-type: none"> <li>● COPYO</li> <li>● Press PROG START</li> </ul>	<p>Name Disk Copy/Dump program uses to refer to output file being created.</p> <p>If you are only printing records from a file, press PROG START instead of typing COPYO. The next keyword prompted will be MODIFY.</p>
UNIT	R1, R2, F1, or F2	Location of disk on which output file is to be created.
PACK	disk name	Name of disk on which output file is to be identified on disk.
LABEL	file name	Name by which output file is to be identified on disk.
RECORDS or TRACKS	number	Size of output file expressed either as number of records (RECORDS) or number of disk tracks (TRACKS).
RETAIN	T, P, or S	Designation (temporary, permanent, or scratch) of output file.
MODIFY	RUN	-----
<p>↑ Only the key-words listed here are required. You can bypass the rest.</p>	<p>↑ You end every response by pressing PROG START.</p>	

## BUILD Sequence for Copying or Printing Files

<i>Keywords</i>	<i>Responses</i>	<i>Considerations</i>
<b>READY</b>	<b>BUILD</b>	-----
<b>BUILD NAME</b>	procedure name	Name by which procedure will be identified in source library.
<b>UNIT</b>	R1, R2, F1, or F2	Location of disk containing source library.
<b>LOAD NAME</b>	\$COPY	Name of Disk Copy/Dump program.
<b>UNIT</b>	R1, R2, F1, or F2	Location of disk containing Disk Copy/Dump program.
<b>FILE NAME</b>	COPYIN	Name Disk Copy/Dump program uses to refer to file to be copied (input file).
<b>UNIT</b>	R1, F1, R2, or F2	Location of disk containing file to be copied.
<b>PACK</b>	disk name	Name of disk containing file to be copied.
<b>LABEL</b>	file name	Name by which file to be copied is identified on disk.
<b>FILE NAME</b>	<ul style="list-style-type: none"> <li>• COPYO</li> <li>• Press PROG START</li> </ul>	<p>Name Disk Copy/Dump program uses to refer to output file being created.</p> <p>If you are only printing records from a file, press PROG START instead of typing COPYO. The next keyword prompted will be MODIFY.</p>
<b>UNIT</b>	R1, R2, F1, or F2	Location of disk on which output file is to be created.
<b>PACK</b>	disk name	Name of disk on which output file is to be created.
<b>LABEL</b>	file name	Name by which output file is to be identified on disk.
<b>RECORDS or TRACKS</b>	number	Size of output file expressed either as number of records (RECORDS) or number of disk tracks (TRACKS).
<b>RETAIN</b>	T, P, or S	Designation (temporary, permanent, or scratch) of output file.
<b>MODIFY</b>	<ul style="list-style-type: none"> <li>• INCLUDE utility control statements RUN</li> <li>• RUN</li> </ul>	<p>Response when including control statements in procedure.</p> <p>Response when not including control statements in procedure.</p>
<p>↑ Only the key-words listed here are required. You can bypass the rest.</p>	<p>↑ You end every response by pressing PROG START.</p>	



## EXAMPLES

### Copying an Entire Disk

```

READY          - LOAD
*****
010   LOAD     NAME      - $COPY
011           UNIT      - F1
020   DATE     (XX/XX/XX) -
030   SWITCH   (00000000) -
040   FILE     NAME      -
*****

```

MODIFY

RUN

#### OCL LOAD Sequence

Circled areas are operator responses.

Keywords for which no responses are shown are the ones bypassed. If you press ENTER— after responding to UNIT, the DATE, SWITCH, and FILE NAME keywords are not prompted.

RUN is the response to MODIFY even though the two words do not appear on the same line.

```

ENTER '//' CONTROL STATEMENT
// COPYPACK FROM-F2,TO-R2
ENTER '//' CONTROL STATEMENT
// END
COPYPACK IS COMPLETE

```

Message printed by Disk Copy/Dump program.

Control statement supplied by operator.

System reprompts. END statement terminates sequence.

Message printed by Disk Copy/Dump program to indicate successful copy.

#### Explanation

- The Disk Copy/Dump program is loaded from the fixed disk on drive 1 (UNIT-F1 in OCL sequence).
- The contents of the fixed disk on drive 2 (FROM-F2 in COPYPACK statement) is copied onto the removable disk on drive (TO-R2).

# Copying a File From One Disk to Another

```

READY          - LOAD
*****
010  LOAD      NAME      - $COPY
011                UNIT  - F1
020  DATE
030  SWITCH
040  FILE      NAME      - COPYIN
041                UNIT  - F1
042                PACK   - A1
043                LABEL  - MASTER
050  FILE      NAME      - COPYO
051                UNIT  - R1
052                PACK   - B2
053                LABEL  - BACKUP
054                RECORDS -
055                TRACKS - 50
056                LOCATION -
057                RETAIN - P
*****
MODIFY
RUN
  
```

File to be copied (input file)




### OCL LOAD Sequence

Circled areas are operator responses.  
 Keywords for which no responses are shown are the ones bypassed.  
 RUN is the response to MODIFY even though the two words do not appear on the same line.

File being created (output file)

```

ENTER '/' CONTROL STATEMENT
// COPYFILE OUTPUT-DISK
// END
  
```

-  Message printed by Disk Copy/ Dump program.
-  Control statement supplied by operator.
-  System reprompts. END statement terminates sequence.

## Explanation

- Disk Copy/Dump program is loaded from fixed disk on drive 1 (UNIT-F1 in OCL sequence).
- Input file (OCL sequence):
  1. Name that identifies file on disk is MASTER (LABEL-MASTER).
  2. Disk that contains the file is the fixed disk 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 removable disk on drive 1 (UNIT-R1). Its name is B2 (PACK-B2).
  3. The file is to be permanent (RETAIN-P).
  4. The length of the file is 50 tracks (TRACKS-50).
- The COPYFILE statement tells the program to create the output file using all the data from the input file. The output file is a copy of the input file.

## Printing Part of a File

```

READY          * LOAD
*****
010   LOAD   NAME   - $COPY
011           UNIT  - F1
020   DATE           -
030   SWITCH        -
040   FILE   NAME   - COPYIN
041           UNIT  - R1
042           PACK  - B2
043           LABEL - BACKUP
050   FILE   NAME   -
*****
MODIFY          -
RUN
  ENTER '///' CONTROL STATEMENT
// COPYFILE OUTPUT-PRINT
  ENTER '///' CONTROL STATEMENT
// SELECT KEY, FROM-'ADAMS',TO-'BAKER"
  ENTER '///' CONTROL STATEMENT
// END
  
```

### OCL LOAD Sequence.

Circled areas are operator responses.

Keywords for which no responses are shown are the ones bypassed.

RUN is the response to MODIFY even though the two words do not appear on the same line.

Message printed by Disk Copy/Dump program.

Control statement supplied by operator.

Sequence repeats until operator enters END statement.

### Explanation

- Disk Copy/Dump program is loaded from the fixed disk on drive 1 (UNIT-F1 in OCL sequence).
- Input file (OCL sequence):
  1. Name that identifies the file on disk is BACKUP (LABEL-BACKUP).
  2. Disk that contains the file is the removable disk on drive 1 (UNIT-R1). Its name is B2 (PACK-B2).
- 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).

# LIBRARY MAINTENANCE PROGRAM

The Library Maintenance program (\$MAINT) has four functions:

Function	Meaning
Allocate	Create (reserve space for), delete, reorganize, and change the sizes of libraries.
Copy	Place entries in, and display the contents of, libraries.
Delete	Delete library entries.
Rename	Change the names of library entries.

The control statements you must supply depend on the function you are using.

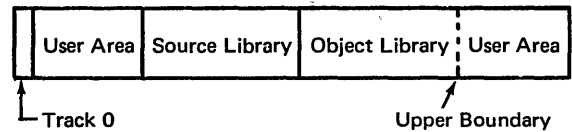
## Library Description

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 further translation before being loaded for execution. (They are sometimes called nonexecutable object programs.)

## Location of Libraries on Disk

Libraries can be located anywhere on disk. However, 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

Entries are stored in the object library serially; that is, a twenty-sector program occupies twenty consecutive sectors. Temporary entries follow all permanent entries in the object library. This occurs because a permanent entry causes all temporary entries to be deleted. The permanent entry is then loaded into the first available space large enough to hold it. This is usually the space following the last permanent entry in the library.

If necessary, the upper boundary is changed to allow more space for temporary entries. But when a permanent entry is placed in the library, 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 a permanent entry is deleted and replaced with a permanent entry using fewer sectors. The Library Maintenance program scans the library to see what sectors are available. The entry is then placed into the gap that has the fewest sectors over and above the number required by the new entry. If the entry is the same size, no sectors are lost.

If the number of unusable sectors becomes excessive, the library should be reorganized. In reorganizing entries, the Library Maintenance program shifts entries so that gaps do not appear between them. This makes more sectors available for use.

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, blanks and duplicate characters are removed from entries. Later, if the entries are printed or punched, the blanks and duplicate characters are reinserted.

### 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 libraries and their directories.

### Organization of This Section

The four functions are described separately. Every description contains the following:

1. List of specific uses.
2. Control statement summary indicating the form of the control statement needed for each use.
3. Parameter summary explaining the contents and meanings of control-statement parameters.
4. Parameter descriptions explaining, in detail, the contents and meanings of the parameters. Because delete and rename are not complex functions, those functions do not need this type of description. The parameter summaries are sufficient.
5. Examples that include OCL statements, utility control statements, and explanations of their use.

OCL considerations for the program precede the examples.

# ALLOCATE

## Uses

- Create (reserve space for) libraries.
- Change the sizes of libraries.
- Delete libraries.
- Reorganize libraries.

## CONTROL STATEMENT SUMMARY

```
// ALLOCATE TO-code, SOURCE- $\left\{ \begin{matrix} \text{number} \\ R \end{matrix} \right\}$ , OBJECT- $\left\{ \begin{matrix} \text{number} \\ R \end{matrix} \right\}$ , SYSTEM- $\left\{ \begin{matrix} \text{NO} \\ \text{YES} \end{matrix} \right\}$ , WORK-code
```

	— Use —	— Parameters Needed —	
Source Library	Create	TO-code, SOURCE-number, WORK-code	WORK parameter needed only if the disk contains an object library that you are not deleting.
	Change Size	TO-code, SOURCE-number, WORK-code	
	Delete	TO-code, SOURCE-0	
	Reorganize	TO-code, SOURCE-R, WORK-code	— DELETES ALL TEMP.
Object Library	Create	TO-code, OBJECT-number, SYSTEM- $\left\{ \begin{matrix} \text{NO} \\ \text{YES} \end{matrix} \right\}$	
	Change Size	TO-code, OBJECT-number, WORK-code	
	Delete	TO-code, OBJECT-0	
	Reorganize	TO-code, OBJECT-R, WORK-code	— DELETES ALL TEMP.

↑ You can indicate a source-library use, an object-library use, or uses involving both libraries (for example, deleting the source library and changing the size of the object library).

↑ If you are indicating uses for both libraries, use only one TO parameter. (The libraries must be on the same disk.) Also, use only one WORK parameter if both uses require a WORK parameter.

## ALLOCATE PARAMETER SUMMARY

TO-code

Location of disk you are using. Possible codes are R1, F1, R2, and F2.

SOURCE-0

Delete the source library.

SOURCE-number

If disk does not contain a source library, program creates one.  
Number indicates the number of tracks you want to assign.

If disk already contains a source library, program changes its size  
and reorganizes it. (This includes deleting temporary entries.)  
Number indicates the total number of tracks you want in source library.

SOURCE-R

Reorganize the source library. Program deletes temporary entries  
while reorganizing the library.

OBJECT-0

Delete the object library.

OBJECT-number

If disk does not contain an object library, program creates one. Number  
indicates the number of tracks you want to assign.

If disk already contains an object library, program changes its size and  
reorganizes it. (This includes deleting temporary entries.) Number indi-  
cates the total number of tracks you want in object library.

OBJECT-R

Reorganize the object library. Program deletes temporary entries while reor-  
ganizing the library.

SYSTEM-NO

Assign one track to object library directory. Object library directory  
will not be large enough to contain system program entries.

SYSTEM-YES

Assign three tracks to object library directory. Object library directory  
will be large enough to contain system program entries.

WORK-code

Location of disk 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 disk that contains, or will contain, the library. If the program use involves both libraries, the libraries must be on the same disk.

Codes for the possible locations are as follows:

Code	Location
R1	Removable disk on drive 1.
F1	Fixed disk on drive 1.
R2	Removable disk on drive 2.
F2	Fixed disk on drive 2.

### SOURCE Parameter

The SOURCE parameter identifies source-library uses:

Parameter	Use
SOURCE-number (number is not zero)	If the disk contains no source library, parameter means create a source library. Number is the number of tracks you want to assign to the library.  If the disk contains a source library, parameter means change the library size. Number is the number of tracks you want to assign to the library.
SOURCE-0	Delete source library.
SOURCE-R	Reorganize source library.

## Disk Considerations for Creating a Source Library (SOURCE-number)

Number of Source Libraries Allowed: One per disk.

If the disk already contains a source library, the SOURCE-number parameter causes the program to change the library size to the number of tracks indicated in the parameter.

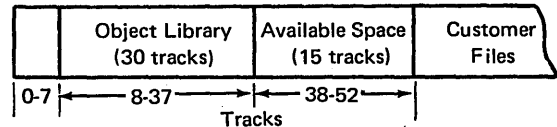
Source Library Size: The minimum size is one track. The maximum is the number of tracks in the available disk area.

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.

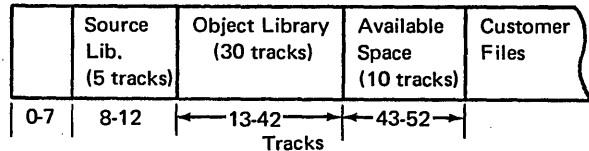
Placement of Source Library (Disk With an Object Library): Source library must immediately precede the object library. Therefore, a disk area large enough for the source library must immediately precede or follow the object library.

If the available disk area follows the object library, the program moves the object library to make room for the source library as the following illustration shows. To do this, it needs a work area (see *WORK Parameter* in this section).

Disk Space Before Source Library:



Disk Space After Source Library:



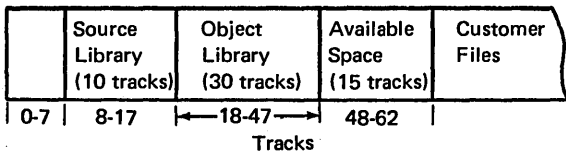
Placement of Source Library (Disk Without an Object Library): Program assigns the source library to the first available disk area large enough for the library.

**Disk Considerations for Changing the Size of a Source Library (SOURCE-number)**

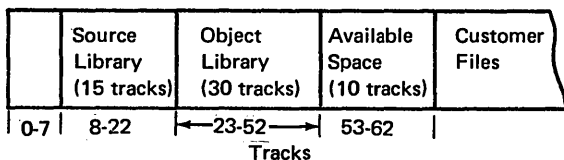
**Making the Source Library Larger:** If the disk contains an object library, space must be available immediately following the object library. The program moves the object library to make tracks available at the end of the source library, as the following illustration shows. The starting location of the source library remains the same.

If the disk doesn't contain an object library, space must be available immediately following the source library.

Disk Before Tracks Are Added to Source Library:

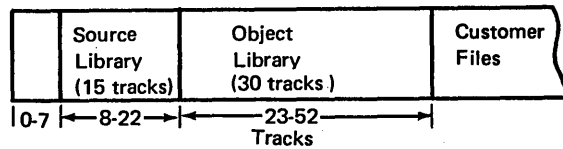


Disk After Five Tracks Are Added to Source Library:

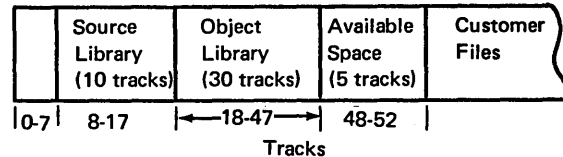


**Making the Source Library Smaller:** If the disk contains an object library, the program moves the end location of the source library to make the library smaller. The starting location remains the same. The program then moves the object library so that no gap appears between the two libraries. Space, therefore, becomes available following the object library, not preceding it, as the following illustration shows:

Disk Before Source-Library Size Was Decreased:



Disk After Five Tracks Were Taken From Source Library:



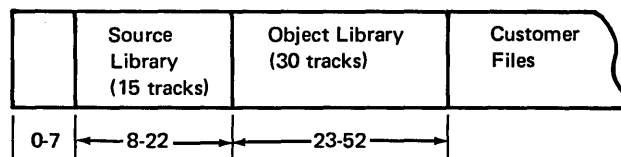
If the disk doesn't contain an object library, the program moves the end location of the source library to make the source library smaller. The starting location remains the same.

**Reorganizing the Source Library:** Any time the program changes the library size, it also reorganizes the library (see *Disk Considerations for Reorganizing a Source Library* in this section). To do this, it needs a work area (see *WORK Parameter* in this section).

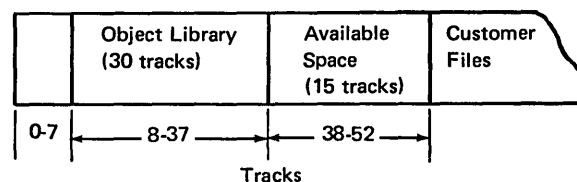
**Disk Considerations for Deleting a Source Library (SOURCE-0)**

The program makes the disk area occupied by the source library available for other use.

Disk Before Source Library Deleted



Disk After Source Library Deleted



**Disk Considerations for 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.

Work Area: The program needs a work area (see WORK Parameter).

**OBJECT Parameter**

The OBJECT parameter identifies object-library uses:

Parameter	Use
OBJECT-number (number not zero)	If the disk doesn't contain an object library, this parameter means create an object library. Number is the number of tracks you want to assign to the library.  If the disk contains an object library, this parameter means change the library size. Number is the number of tracks you want to assign to the library.
OBJECT-0	Delete object library.
OBJECT-R	Reorganize object library.

**Disk Considerations for Creating an Object Library (OBJECT-number)**

Number of Object Libraries Allowed: One per disk. If the disk already contains one, the OBJECT-number parameter causes the program to change the library size to the number of tracks indicated in the parameter.

Object Library Size: Minimum size is 30 tracks if the object library will contain a minimum system; otherwise, the minimum is three tracks. (A minimum system is made up of those system programs necessary to load and run programs; it does not include those programs that generate and maintain a system.) However, if inquiry, the Data Recorder, or the CRT (2265-2) are included in the system, 32 tracks are needed for the system object library.

Maximum size is the number of tracks in the available area.

Library directory: The first three tracks in the library are reserved for the library directory if the library will contain system programs. Otherwise, only the first track is reserved.

Scheduler work area: If the library will contain system programs, the disk area to contain the library must be large enough to also contain a work area for the Scheduler program (one of the system programs). The work area space is not included in the number you specify in the OBJECT parameter. It is calculated and assigned by the Library Maintenance program. The amount of additional space needed depends on the capacity of your system and whether your programming system contains the Inquiry feature:

Capacity	Scheduler Work Area Size	
	No Inquiry	Inquiry
8K bytes	2 tracks	5 tracks
12K bytes	2 tracks	6 tracks
16K bytes	2 tracks	7 tracks

Placement of Object Library (Disk With a Source Library): Space for the object library must be available immediately following the source library.

Placement of Object Library (Disk Without a Source Library): Program assigns the object library to the first available disk area large enough for the library.

#### Disk Considerations for Changing the Size of 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 (see *Disk Considerations for Reorganizing an Object Library* in this section). To do this, it needs a work area (see *WORK Parameter* in this section).

#### Disk Considerations for Deleting an Object Library (OBJECT-0)

**Deleting the Library:** The program makes the disk area occupied by the object library available for other use.

**Restriction:** The Library Maintenance program will not delete either of the following object libraries:

1. The library from which the Library Maintenance program was loaded.
2. The library containing the system programs that are controlling program loading.

#### Disk Considerations for Reorganizing an Object Library (OBJECT-R)

**Reason for Reorganizing the Library:** Gaps can occur between object-library entries when you add and delete entries. By reorganizing the library, you might salvage enough space for additional entries without increasing the size of the library.

**Reorganizing the Library:** Entries are relocated so that no gaps appear between them. All temporary entries are deleted.

**Work Area:** The program needs a work area (see *WORK Parameter* in this section).

#### WORK Parameter

The WORK parameter (WORK-code) indicates the location of the disk that contains a work area. Library entries are temporarily stored in the work area while the program moves and reorganizes libraries.

Codes for the possible disk locations are as follows:

Code	Location
R1	Removable disk on drive 1.
F1	Fixed disk on drive 1.
R2	Removable disk on drive 2.
F2	Fixed disk on drive 2.

## Disk Considerations for Work Area

**Size of the Work Area:** The work area must be large enough to hold the entire source library, object library, or both libraries depending on the program use:

Use <sup>①</sup>	Contents of Work Area
Create a source library (disk contains an object library).	Object library.
Change source library size (disk contains an object library).	Source library and object library.
Change source library size (disk doesn't contain an object library).	Source library.
Reorganize source library (disk contains an object library).	Source library and object library.
Reorganize source library (disk doesn't contain an object library).	Source library.
Change object library size.	Object library.
Reorganize object library.	Object library.

**Location of Work Area on Disk:** The program uses the first available disk area large enough to hold the library, or libraries.

**Location of Disk Containing the Work Area:** The work area can be on either disk on either drive. However, it cannot be the same disk as the one you specified in the TO parameter. The only requirement is that the disk must have an available area large enough for the work area. If your system has two disk drives, the program works faster if the disk containing the libraries is on a different drive than the disk containing the work area.

### SYSTEM Parameter

The SYSTEM parameter applies to creating object libraries. It tells the program whether or not you intend to include system programs in the library.

### Include System Programs

SYSTEM-YES means you intend to include system programs. It causes the Library Maintenance program to do the following:

1. Assign three tracks (instead of one) to the library directory.
2. Assign space for a scheduler work area.

The directory will be large enough for all system programs: those necessary for program loading and running (minimum system), and those necessary for generating and maintaining a system.

Space for the Scheduler work area is assigned immediately preceding the object library. If the disk contains a source library, the work area will appear between the source and object libraries. For information about the size of the work area, see *Disk Considerations for Creating an Object Library* in this section.

### Do Not Include System Programs

SYSTEM-NO means you do not intend to include system programs in the library. The Library Maintenance program does not assign space for the Scheduler work area, and it assigns one track (instead of three) to the library directory. SYSTEM-NO is assumed if you omit the SYSTEM parameter.

<sup>①</sup> 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.

# COPY

## Uses

Reader-to-Disk ①

Add or replace a library entry.

Disk-to-Disk

Copy one library entry.

Copy library entries that have names beginning with certain characters. ②

Copy all library entries. ②

Copy minimum system. ③

Copy an IBM program.

Disk-to-Printer

Print one library entry.

Print library entries that have names beginning with certain characters. ②

Print all library entries of a certain type. ④

Print directory entries for library entries of a certain type. ④

Print entries from all directories including system directory.

Print system directory only.

Disk-to-Card

Punch one library entry.

Punch library entries that have names beginning with certain characters. ②

Punch all library entries of a certain type. ④

**Disk-to-Printer  
and Card**



Print and punch one library entry.

Print and punch library entries that have names beginning with certain characters. ②

Print and punch all library entries of a certain type. ④

- ① The reader is the system input device. The system input device can be either the keyboard or a card reader.
- ② You can specify the following types of entries: source statements, procedures, object programs, routines, or all of these types.
- ③ Minimum system consists of the system programs necessary to load and run programs. It does not include the system programs necessary to generate and maintain the system.
- ④ You can specify one of the following types of entries: source statements, procedures, object programs, or routines.

## Control Statement Summary: Reader-To-Disk

### Add or Replace a Library Entry (Reader is Keyboard) ①

```
// COPY FROM-READER,LIBRARY- $\left\{ \begin{array}{c} S \\ P \\ O \\ R \end{array} \right\}$ ,NAME-name,TO-code,RETAIN- $\left\{ \begin{array}{c} T \\ P \\ R \end{array} \right\}$   
Library Entry  
// CEND
```

### Add or Replace a Library Entry (Reader is Card Reader) ②

```
// COPY FROM-READER,LIBRARY- $\left\{ \begin{array}{c} S \\ P \\ O \\ R \end{array} \right\}$ ,NAME-name,TO-code,RETAIN- $\left\{ \begin{array}{c} T \\ P \\ R \end{array} \right\}$   
Library Entry  
// CEND
```

① // COPY statement, library entry, and // CEND statement are all read from the keyboard.

② // COPY statement, library entry, and // CEND statement are all read from cards.

## Control Statement Summary: Disk-To-Disk

### Copy One Library Entry

```
// COPY FROM-code,LIBRARY- $\left\{ \begin{array}{c} S \\ P \\ O \\ R \end{array} \right\}$ ,NAME-name,TO-code,RETAIN- $\left\{ \begin{array}{c} T \\ P \\ R \end{array} \right\}$ ,NEWNAME-name ①
```



### Copy Library Entries That Have Names Beginning With Certain Characters

```
//COPY FROM-code,LIBRARY- $\left. \begin{array}{c} S \\ P \\ O \\ R \\ \hline ALL \end{array} \right\}$ ,NAME-characters.ALL,TO-code,RETAIN- $\left. \begin{array}{c} T \\ P \\ R \end{array} \right\}$ ②,NEWNAME-characters③
```

### Copy All Library Entries

```
//COPY FROM-code,LIBRARY- $\left. \begin{array}{c} S \\ P \\ O \\ R \\ \hline ALL \end{array} \right\}$ ,NAME-ALL,TO-code,RETAIN- $\left. \begin{array}{c} T \\ P \\ R \end{array} \right\}$ ②
```

### Copy Minimum System

```
//COPY FROM-code,LIBRARY-O,NAME-SYSTEM,TO-code
```

### Copy an IBM Program

```
// COPY FROM-code,LIBRARY-O,NAME-$cc.ALL,TO-code,RETAIN- $\left. \begin{array}{c} T \\ P \end{array} \right\}$ 
```

- ① NEWNAME parameter is needed in either of the following cases:
1. If you want the copy to have a different name than the original entry.
  2. If you want to replace an entry on the TO disk with an entry from the FROM disk, but the entries have different names.
- ② If you use T or P in the RETAIN parameter, the TO library entries will have the T or P retain type. If you use R, the TO library entries will have the same retain type that they had in the FROM library.
- ③ Use the NEWNAME parameter only if you want the names of the copies to begin with different characters than the names of the original entries. The number of characters must equal the number of characters in the NAME parameter.

## Control Statement Summary: Disk-To-Printer

### Print One Library Entry

```
// COPY FROM-code, LIBRARY- $\left. \begin{array}{c} S \\ P \\ O \\ R \end{array} \right\}$ , NAME-name, TO-PRINT
```

### Print Library Entries That Have Names Beginning With Certain Characters

```
// COPY FROM-code, LIBRARY- $\left. \begin{array}{c} S \\ P \\ O \\ R \\ ALL \end{array} \right\}$ , NAME-characters.ALL, TO-PRINT
```

### Print All Library Entries of a Certain Type

```
// COPY FROM-code, LIBRARY- $\left. \begin{array}{c} S \\ P \\ O \\ R \end{array} \right\}$ , NAME-ALL, TO-PRINT
```

### Print Directory Entries for Library Entries of a Certain Type

```
// COPY FROM-code, LIBRARY- $\left. \begin{array}{c} S \\ P \\ O \\ R \end{array} \right\}$ , NAME-DIR, TO-PRINT
```

### Print Entries From All Directories Including System Directory

```
// COPY FROM-code, LIBRARY-ALL, NAME-DIR, TO-PRINT
```

### Print System Directory Only

```
// COPY FROM-code, LIBRARY-SYSTEM, NAME-DIR, TO-PRINT
```

## Control Statement Summary: Disk-To-Card

### Punch One Library Entry

```
// COPY FROM-code,LIBRARY- $\left. \begin{array}{c} S \\ P \\ O \\ R \end{array} \right\}$ ,NAME-name,TO-PUNCH
```

### Punch Library Entries That Have Names Beginning With Certain Characters

```
// COPY FROM-code,LIBRARY- $\left. \begin{array}{c} S \\ P \\ O \\ R \\ ALL \end{array} \right\}$ ,NAME-characters.ALL,TO-PUNCH
```

### Punch All Library Entries of a Certain Type

```
// COPY FROM-code,LIBRARY- $\left. \begin{array}{c} S \\ P \\ O \\ R \end{array} \right\}$ ,NAME-ALL,TO-PUNCH.
```

## Control Statement Summary: Disk to Printer and Card

### Print and Punch One Library Entry

```
//COPY FROM-code,LIBRARY- $\left. \begin{array}{c} S \\ P \\ O \\ R \end{array} \right\}$ ,NAME-name,TO-P RTPCH
```

### Print and Punch Library Entries That Have Names Beginning With Certain Characters

```
//COPY FROM-code,LIBRARY- $\left. \begin{array}{c} S \\ P \\ O \\ R \\ ALL \end{array} \right\}$ ,NAME-characters.ALL,TO-P RTPCH
```

### Print and Punch All Library Entries of a Certain Type

```
//COPY FROM-code,LIBRARY- $\left. \begin{array}{c} S \\ P \\ O \\ R \end{array} \right\}$ ,NAME-ALL,TO-P RTPCH
```

## Parameter Summary

**FROM-READER**

Entry to be placed in library is to be read from system input device which can be a keyboard or card reader.

**FROM-code**

Location of disk containing library entries being copied, printed, or punched. Possible location codes are:

Code	Meaning
R1	Removable disk on drive 1.
F1	Fixed disk on drive 1.
R2	Removable disk on drive 2.
F2	Fixed disk on drive 2.

**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, 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 <sup>①</sup>	Only those entries beginning with the indicated characters (you can use up to five characters).
ALL <sup>①</sup>	All entries (of the type indicated in LIBRARY parameter).

① On a disk-to-disk copy, check that sufficient space has been allocated on the 'TO' disk.

NAME-SYSTEM

Only system programs that make up the minimum system are involved in the copy use.

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.

NAME-\$cc.ALL

The IBM program with the name beginning with the indicated characters (\$cc) is involved in the copy use. For example, \$MA.ALL means the Library Maintenance program (\$MAINT).

RETAIN- { T }  
          { P }  
          { R }

*Adding Entry to Library.* RETAIN gives designation of entry:

Code	Meaning
T	Temporary.
P or R	Permanent.

*Replacing Existing Library Entry.* RETAIN gives designation of 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	Use same designation as existing entry. Do not halt before replacing entry.

*Printing or Punching Entries.* RETAIN parameter is ignored:

**TO-code**

Location of disk that is to contain the copies of the entries:

Code	Meaning
R1	Removable disk on drive 1.
F1	Fixed disk on drive 1.
R2	Removable disk on drive 2.
F2	Fixed disk on drive 2.

**TO-PRINT**

Entries are being printed.

**TO-PUNCH**

Entries are being punched.

**TO-PRTPCH**

Entries are being printed and punched.

**NEWNAME-name**

Name you want used on the TO disk to identify the entries being put on that disk. 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 TO disk. 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.

## Parameter Descriptions

### FROM and TO Parameters

The FROM parameter identifies the device from which the program will copy library or directory entries. The TO parameter identifies the destination of these entries. Together, the parameters define the copy function being done:

	Parameters
Reader-to-disk	FROM-READER TO-code
Disk-to-disk	FROM-code TO-code
Disk-to-printer	FROM-code TO-PRINT
Disk-to-cards	FROM-code TO-PUNCH
Disk-to-printer and cards	FROM-code TO-PRTPCH

The codes indicating the possible disk locations are as follows:

Disk Code	Meaning
R1	Removable disk on drive 1.
F1	Fixed disk on drive 1.
R2	Removable disk on drive 2.
F2	Fixed disk on drive 2.

### Reader-to-Disk Considerations

Input: The program reads one library entry. It can be any one of the following types:

1. Source statements.
2. Procedure.
3. Object program.
4. Routine.

Output: Blanks are removed from source statements before they are put in the source library.

Procedures are put in the source library in the form in which the program reads them. The program does not check them for errors.

Object programs and routines are placed in the object library after being compressed.

System Input Device: The entry is read from the system input device, which is normally the keyboard. The operator can, however, change the system input device to a card reader during initial program loading (IPL) using the OCL READER statement.

Replacing Existing Entries: The program can replace an existing library entry with the entry you are putting in the library.

The program can halt before replacing an existing entry. Whether or not it does depends on the RETAIN parameter you use. See *RETAIN Parameter* in this section for more information.

### Disk-to-Disk Considerations

Input: The program can copy one or more library entries from one disk to another. The types of entries can be:

1. Source statements.
2. Procedures.
3. Object programs.
4. Routines.
5. All of the preceding types.
6. Minimum system.

See *LIBRARY Parameter* in this section.



**Output:** The entries, regardless of their type, are copied from one disk to the other without change.

**Disks:** The disk from which the entries are copied and the disk to which the entries are copied must be different disks.

If you are copying a minimum system, the disk you indicate in the TO parameter must not already contain the minimum system.

**Replacing Existing Entries:** The program can replace existing library entries with the entries you are putting in the library. (See *NAME Parameter* and *NEWNAME Parameter* in this section.)

The program can halt before replacing an existing entry. Whether or not it does depends on the *RETAIN* parameter. See *RETAIN Parameter* in this section for more information.

#### **Disk-to-Printer Considerations**

**Types of Entries That Can Be Printed:** The program can print one or more library entries. They can be any of the following types:

1. Source statements.
2. Procedures.
3. Object programs.
4. Routines.
5. All of the preceding types (limited to entries having the same name and entries beginning with the same characters).

The program can print the following types of directory entries.

1. Source statements.
2. Procedures.
3. Object programs.
4. Routines.
5. System directory.
6. All types.

The program will print out sorted names only if three tracks are left available as a work area.

**Printout of Library Entries:** Blanks and duplicate characters are reinserted into source statements to make them readable.

Procedures, object programs, and routines are printed as they exist in the library.

**Printout of Directory Entries:** The following three illustrations show the three types of directory printouts.

**Source Library Directory**

**PRINTOUT**

SOURCE DIRECTORY FROM XX VOL. ID XXXXXX

TYPE	NAME	ADDRESS		ATTR	NO. OF SECTORS
		FIRST	LAST		
X	XXXXXX	TTT-SS	TTT-SS	X	XXX

**EXPLANATION**

Heading	Meaning
TYPE	S=source statements P=procedure
NAME	Name of library entry (up to six 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.
ATTR (Attribute)	T=temporary P=permanent
NO. OF SECTORS	Total number of sectors used for the library entry.

## Object Library Directory

PRINTOUT

OBJECT LIBRARY FROM XX<sup>①</sup> VOL. ID XXXXXX<sup>②</sup>

TYPE <sup>③</sup>	NAME	DISK ADDRESS	CYL/SEC <sup>④</sup>	TXT SEC <sup>④</sup>	LINK ADDR	RLD DISP	ENTRY POINT	CORE SEC	ATTR	LEVEL	TOTAL SEC
A L	XXXXXX	TTT/SS	CC/SS	XXX	XXXX	XX	XXXX	XXX	XXXX	XXX	XXXXX

### EXPLANATION

#### Heading

#### Meaning

TYPE

A } P=permanent  
 T=temporary  
 L } O=object  
 R=routine

NAME

Name of library entry (up to six characters)

DISK ADDRESS

Address where library entry begins on disk. EXAMPLE: 015/10 means track 15, sector 10 (in decimal).

CYL/SEC

Address where library entry begins on disk (in hexadecimal).

TXT SEC

Object programs only. It indicates the number of sectors used for text portion of library entry. Object library entries are made up of two parts: text and RLD. Text is the program or routine. RLD is information used in loading the program or routine for execution.

LINK ADDRESS

Object programs only. Assigned core address of this library entry.

RLD DISP

Object programs only. It indicates the 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 is in the next sector.

ENTRY POINT

Object programs only. Main storage address where program execution begins (includes relocations).

CORE SECTORS

Core size, given in sectors, required to run the program.

① Disk where the directory was printed from.

② Volume ID from the disk.

③ A=Attribute,L=Library.

④ T=Track,S=Sector,C=Cylinder

**ATTRIBUTES**

**Byte 1:**

Bit 0=1—permanent

Bit 0=0—temporary

Bit 1=1—inquiry

Bit 2=1—inquiry evoking

Bit 3=1—must run dedicated

Bit 4=1—requires source information

Bit 5=1—deferred mounting allowed

Bit 6=1—PTF applied

Bit 7=1—No AUTOLINK requested

**Byte 2:**

Bit 0=1—Sysin required for execution

Bits 1-7—reserved

**LEVEL**

Release level

**TOTAL SECTORS**

Total number of disk sectors occupied by the library entry

## System Directory Printout

SYSTEM DIRECTORY FROM XX VOL. ID XXXXXX

### SOURCE LIBRARY

Source Directory Location	TTT-SS
Next Available Library Sector	TTT-SS
End of Library	TTT-SS
Number of Directory Sectors	XXX
Number of Permanent Library Sectors	XXX
Number of Active Library Sectors	XXX
Number of Available Library Sectors	XXX
Allocated Size of Library	YYY

### OBJECT LIBRARY

Object Directory Location	TTT-SS
End of Directory	TTT-SS
Start of Library	TTT-SS
Allocated End of Library	TTT-SS
Extended End of Library	TTT-SS
Number of Available Permanent Directory Entries	XXX
Number of Available Temporary Directory Entries	XXX
First Temporary Directory Entry	TTT-SS-DDD
Next Available Temporary Directory Entry	TTT-SS-DDD
Next Available Library Sector for Permanents	TTT-SS
Next Available Library Sector for Temporaries	TTT-SS
Number of Available Library Sectors for Permanents	XXX
Number of Available Library Sectors for Temporaries	XXX
Number of Active Library Sectors	XXX
Number of Active O. Permanent Library Sectors	XXX
Number of Active R. Permanent Library Sectors	XXX
Allocated Size of Library	YYY
Roll-in/Roll-out Location	TTT-SS
Roll-in/Roll-out Size	YYY
SWA Location	TTT-SS
SWA Size	YYY
Start of Libraries	TTT-SS
End of Libraries	TTT-SS

TTT-SS-DDD means track, sector, and displacement. Displacement is the number of characters from the beginning of the sector. XXX = number of sectors. YYY = number of tracks.

### Disk-to-Card Considerations

#### Type of Entries That Can Be Punched

The program can punch one or more of the following types of entries into cards:

1. Source statements.
2. Procedures.
3. Object programs.
4. Routines.
5. All types limited to those beginning with the same characters.

#### Form of Library Entries in Cards

Blanks and duplicate characters are reinserted into source statements to make the statements readable.

Procedures, object programs, and routines are punched as they appear in the library.

#### COPY Statement

A card containing a partially completed COPY statement (// COPY FROM-READER) is punched preceding each library entry. It identifies the beginning of the entry. If you copy the entry from cards to disk (reader-to-disk), be sure to complete the COPY statement.

#### CEND Statement

A card containing a CEND control statement (//CEND) is punched immediately after each entry. You need it for copying the entry from cards to disk at a later time.

#### Disk-to-Printer-and-Card Considerations

The considerations for disk-to-printer-and-card are the same as disk-to-printer and disk-to-card with this exception: you cannot print and punch directory entries.

### LIBRARY Parameter

The LIBRARY parameter identifies the type of library or directory entries involved in the copy use. Codes for the possible types are as follows:

Code	Meaning
S	Source statements (source library).
P	OCL procedures (source library).
O	Object programs (object library).
R	Routines (object library).
All	All types from both libraries (S, P, O, and R).
SYSTEM	System directory entries.

#### Types of Library Entries

**Source Library:** Source statements can be any combination of valid System/3 characters. Examples of source statements are RPG source programs and sequence specifications for the Disk Sort program.

Procedures are sets of OCL statements. Procedures for utility programs can include control statements following the OCL statements.

**Object Library:** Object programs are programs and subroutines in such a form that they can be loaded for execution. They are sometimes called executable programs.

Routines are programs and subroutines that need further translation before being loaded for execution. They are sometimes called nonexecutable object programs.

## Types of Directory Entries

**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 *Disk-to-Printer Considerations* in this section.

The Library Maintenance program makes entries in the directories when it puts entries in the libraries.

**System Directory:** Every disk 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 *Disk-to-Printer Considerations* in this section.

The Library Maintenance program creates and maintains the system directory.

## NAME Parameter

The NAME parameter identifies specific entries, of the type indicated in the LIBRARY parameter, involved in the copy use. The information possible in the NAME parameter is as follows:

Information	Meaning
name	Name of the library entry to be copied.
character.ALL	Only those entries beginning with the indicated characters. (You can use up to five characters.)
ALL	All entries (of the type indicated in LIBRARY parameter).
SYSTEM	Only system programs that make up the minimum system.

DIR

Directory entries for all library entries of the type indicated in LIBRARY parameter.

\$cc.ALL

IBM program with the name beginning with the indicated three characters (\$cc). For example: \$MA.ALL refers to Library Maintenance program (SMOINT).

## Naming Entries

**Characters to Use:** Use any combination of System/3 characters except blanks and periods (.). (Appendix J lists the characters.) The names of all 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.

**Length of Name:** The name can be from one to six characters long.

**Restricted Names:** Do not use the names ALL, DIR, and SYSTEM. They have special meanings in the NAME parameter.

## Using Names

**NAME-name:** For reader-to-disk uses, the entry you put in the library will be identified by the name you give in the NAME parameter.

For disk-to-disk uses, the name you give in the NAME parameter identifies the entry being copied (the one on the FROM disk). It will also identify the copy (the one on the TO disk) unless you use a NEWNAME parameter. See *NEWNAME Parameter* in this section.

**NAME-characters.ALL:** The NAME parameter identifies the entries being copied from the FROM disk. The names of the copies and original entries will be the same unless you use a NEWNAME parameter (NEWNAME-characters.ALL). See *NEWNAME Parameter* in this section.

**NAME-ALL (Disk-to-Disk):** The names by which the entries are identified on the FROM disk will also be used on the TO disk to identify the entries.

**NAME-SYSTEM.** The NAME parameter indicates that the minimum system is being 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 are not included.

**NAME-\$cc.ALL:** The NAME parameter indicates that an IBM program is being copied. The names of all IBM programs begin with dollar sign (\$) and are unique within the first three characters.

#### **RETAIN Parameter**

The RETAIN parameter supplies the RETAIN code for the TO disk. It also indicates whether you want the program to halt before replacing existing entries in the library. When printing and/or punching library entries, the RETAIN parameter is not needed. If the RETAIN parameter is not supplied when copying disk-to-disk or reader-to-disk, RETAIN-T is assumed; except when the parameters LIBRARY-ALL and NAME-ALL or LIBRARY-O and NAME-SYSTEM are used, RETAIN-P is assumed.

Codes for the parameter are as follows:

Code	Meaning
T	Only temporary entries. Halt before replacing entries.
P	Only permanent entries. Halt before replacing entries.
R	Both permanent and temporary entries. Do not halt before replacing entries.

If you omit the parameter, RETAIN-T is assumed.

**Temporary Entries:** Temporary entries are entries you don't 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 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 deleted every time you add a permanent entry.

You can use temporary entries as often as you like until they are deleted.



**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 will 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 RETAIN Parameter

If you use RETAIN-T or RETAIN-P, the designation of the entries you are putting in the library is taken from the RETAIN parameter. There is one restriction: a temporary entry cannot replace a permanent entry.

RETAIN-R is normally used to bypass program halts when you are replacing several existing entries, perhaps with updated versions. If you use RETAIN-R and the library does not contain an entry corresponding to the entry you are putting in the library (same type and same name), the program adds the new entry to the library and gives it a permanent designation.

#### NEWNAME Parameter

The NEWNAME parameter applies to disk-to-disk copy uses only. With it, you can give the copies different names than the original entries. The information you can supply in the parameter is as follows:

Information	Meaning
name	Name you want to use for the copy, or copies.
characters.ALL	Beginning characters you want to use in names identifying the copies.

#### Considerations for Using NEWNAME Parameter

**Copy Does Not Replace an Existing Entry:** You can omit the NEWNAME parameter. If you do, the name used for the copy will be taken from the NAME parameter. (The copy will have the same name as the original entry.)

If you use a NEWNAME parameter, follow these rules to construct the name:

1. You can use any System/3 characters except blanks, commas, quotes (apostrophes), and periods. (Appendix J lists the characters.) However, the names of all 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.
2. You can use up to six characters, but do not use the names ALL, DIR, and SYSTEM. They have special meanings in the NAME parameter.

**Copy Replaces an Existing Entry:** If the entry you are copying (the entry on the disk you identify in the FROM parameter) has the same name as the entry you are replacing (the entry on the disk you identify in the TO parameter), you can omit the NEWNAME parameter. If the names are not the same, you must use a NEWNAME parameter to give the name of the entry you are replacing.

## DELETE

### Uses

- Delete an entry from a library.
- Delete temporary or permanent library entries that have names beginning with certain characters. (You can specify the following types of entries: source statements, procedures, object programs, routines, or all of these types.)
- Delete all temporary or permanent library entries of a certain type. (You can specify the following types of entries: source statements, procedures, object programs, or routines.)

## Control Statement Summary

### Delete Library Entry

```
// DELETE FROM-code, LIBRARY-  $\left\{ \begin{array}{c} S \\ P \\ O \\ R \end{array} \right\}$ , NAME-name, RETAIN-  $\left\{ \begin{array}{c} T \\ P \end{array} \right\}$ 
```

### Delete Temporary or Permanent Entries With Names Beginning With Certain Characters

```
// DELETE FROM-code, LIBRARY-  $\left\{ \begin{array}{c} S \\ P \\ O \\ R \\ ALL \end{array} \right\}$ , NAME-characters.ALL, RETAIN-  $\left\{ \begin{array}{c} T \\ P \end{array} \right\}$ 
```

### Delete All Temporary or Permanent Entries of a Certain Type

```
// DELETE ① FROM-code ②, LIBRARY-  $\left\{ \begin{array}{c} S \\ P \\ O \\ R \end{array} \right\}$ , NAME-ALL, RETAIN-  $\left\{ \begin{array}{c} T \\ P \end{array} \right\}$ 
```

- ① For LIBRARY-O, RETAIN-T all temporary entries will be deleted from the object library (O and R). New temporary entries cannot replace deleted temporary entries unless all old temporary entries are deleted. RETAIN-T, NAME-ALL, and LIBRARY-O can be used to delete all temporary entries.
- ② LIBRARY-O, NAME-ALL, RETAIN-P cannot be used if the unit specified in the FROM parameter is the same disk on the LOAD statement or the system pack.

# Parameter Summary

FROM-code

Location of disk that contains library entries you are deleting. Possible codes are:

<i>Code</i>	<i>Meaning</i>
R1	Removable disk on drive 1.
F1	Fixed disk on drive 1.
R2	Removable disk on drive 2.
F2	Fixed disk on drive 2.

LIBRARY- { S P O R }

Type of entries being deleted. Possible codes are:

<i>Code</i>	<i>Meaning</i>
S	Source statements (source library).
P	Procedures (source library).
O	Object programs (object library).
R	Routines (object library).

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:

<i>Code</i>	<i>Meaning</i>
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 five 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: T stands for temporary, and P for permanent.

## RENAME

### Use

Change the name of a library entry.

### Control Statement Summary

```
// RENAME FROM-code,LIBRARY- $\left. \begin{array}{c} \text{S} \\ \text{P} \\ \text{O} \\ \text{R} \end{array} \right\}$ ,NAME-name,NEWNAME-name
```

## Parameter Summary

FROM-code

Location of disk that contains the entry you are renaming.  
Possible codes are:

<i>Code</i>	<i>Meaning</i>
R1	Removable disk on drive 1.
F1	Fixed disk on drive 1.
R2	Removable disk on drive 2.
F2	Fixed disk on drive 2.

LIBRARY- $\left\{ \begin{array}{c} S \\ P \\ O \\ R \end{array} \right\}$

Type of library entry you are renaming. Possible codes are:

<i>Code</i>	<i>Meaning</i>
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.

NEWNAME-name

New name you want to give the entry. Follow these rules to construct the name:

1. You can use any System/3 characters except blanks, commas, quotes (apostrophes), and periods (Appendix J lists the characters). However, the names of all 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.
2. You can use up to six characters, but do not use the names ALL, DIR, and SYSTEM. They have special meanings in the NAME parameter.

## OCL CONSIDERATIONS

<b>LOAD Sequence</b>		
<i>Keywords</i>	<i>Responses</i>	<i>Considerations</i>
READY	LOAD	-----
LOAD NAME	\$MAINT	Name of Library Maintenance program.
UNIT	R1, R2, F1, or F2	Location of disk containing Library Maintenance program.
MODIFY	RUN	-----
↑ Only the key-words listed here are required. You can bypass the rest.	↑ You end every response by pressing PROG START.	

<b>BUILD SEQUENCE</b>		
<i>Keywords</i>	<i>Responses</i>	<i>Considerations</i>
READY	BUILD	-----
BUILD NAME	procedure name	Name by which procedure will be identified in source library.
UNIT	R1, R2, F1, or F2	Location of disk containing source library.
LOAD NAME	\$MAINT	Name of Library Maintenance program.
UNIT	R1, R2, F1, or F2	Location of disk containing Library Maintenance program.
MODIFY	<ul style="list-style-type: none"> <li>• INCLUDE utility control statements</li> <li>RUN</li> <li>• RUN</li> </ul>	Response when including control statements in procedure.  Response when not including control statements in procedure.
↑ Only the key-words listed here are required. You can bypass the rest.	↑ You end every response by pressing PROG START.	

## ALLOCATE EXAMPLES

### Creating Both Source and Object Libraries on a Disk

```

READY          - LOAD
*****
010   LOAD   NAME   - $MAINT
011           UNIT   - F1
020   DATE           -
030   SWITCH         -
040   FILE   NAME   -
*****

MODIFY         -
RUN
  ENTER '// ' CONTROL STATEMENT
// ALLOCATE TO-R1,SOURCE-12,OBJECT-45,SYSTEM-YES
  ENTER '// ' CONTROL STATEMENT
// END
  
```

**OCL LOAD Sequence**  
 Circled areas are operator responses.  
 Keywords for which no responses are shown are the ones bypassed.  
 RUN is the response to MODIFY even though the two words do not appear on the same line.

Message printed by Library Maintenance program.

Control statement supplied by operator.

Program creates libraries, then asks for another control statement.

END statement, supplied by operator, ends the program.

#### Explanation

- Library Maintenance program is loaded from the fixed disk on drive 1 (UNIT-F1 in OCL sequence).
- Libraries are being created on the removable disk on drive 1 (TO-R1 in ALLOCATE statement).
- Source library space is twelve tracks long (SOURCE-12).
- Object library space is 45 tracks long (OBJECT-45). The object library will contain system programs (SYSTEM-YES). Thus, the disk area will also include space for the Scheduler work area.



## Changing the Size of a Source Library

```

READY          - LOAD
*****
010   LOAD     NAME  - $MAINT
011           UNIT  - F1
020   DATE
030   SWITCH
040   FILE     NAME  -
*****

MODIFY         -
RUN
  ENTER '// ' CONTROL STATEMENT
// ALLOCATE TO-R1, SOURCE-15, WORK-F1
  ENTER '// ' CONTROL STATEMENT

// END
  
```

### OCL LOAD Sequence.

Circled areas are operator responses.

Keywords for which no responses are shown are the ones bypassed.

RUN is the response to MODIFY even though the two words do not appear on the same line.

▶ Message printed by Library Maintenance program.

▶ Control statement supplied by operator.

▶ Program changes size of library, then asks for another control statement.

▶ END statement, supplied by operator, ends the program.

### Explanation

- Library Maintenance program is loaded from the fixed disk on drive 1 (UNIT-F1 in OCL sequence).
- Source library is located on the removable disk 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 fixed disk on drive 1 (WORK-F1).

## Deleting the Object Library From a Disk

```

READY                                - LOAD
*****
010  LOAD  NAME  - $MAINT
011          UNIT - F1
020  DATE
030  SWITCH
040  FILE  NAME  -
*****

MODIFY
RUN
  ENTER '// ' CONTROL STATEMENT
// ALLOCATE TO-R1,OBJECT-0
  ENTER '// ' CONTROL STATEMENT

// END

```

### OCL LOAD Sequence.

Circled areas are operator responses.

Keywords for which no responses are shown are the ones bypassed.

RUN is the response to MODIFY even though the two words do not appear on the same line.

Message printed by Library Maintenance program.

Control statement supplied by operator.

Program deletes library, then asks for another control statement.

END statement, supplied by operator, ends the program.

### Explanation

- Library Maintenance program is loaded from the fixed disk on drive 1 (UNIT-F1 in OCL sequence).
- Object library is located on the removable disk 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, the program also deletes the work area.

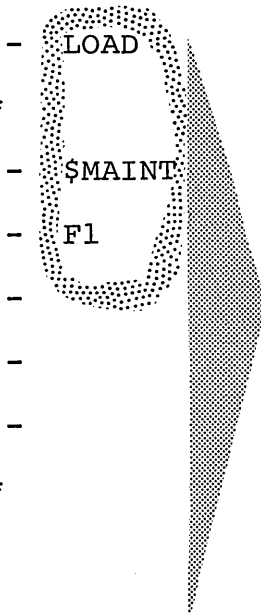
# COPY EXAMPLES

## Copying Minimum System from One Disk to Another

```

READY
*****
010  LOAD  NAME  -
011          UNIT -
020  DATE
030  SWITCH
040  FILE  NAME -
*****

```



### OCL LOAD Sequence

Circled areas are operator responses.

Keywords for which no responses are shown are the ones bypassed.

RUN is the response to MODIFY even though the two words do not appear on the same line.

MODIFY

**RUN**

ENTER '// ' CONTROL STATEMENT

Message printed by Library Maintenance program.

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

Control statement supplied by the operator.

ENTER '// ' CONTROL STATEMENT

Program copies programs, then asks for another control statement.

// END

END statement, supplied by operator, ends the program.

### Explanation

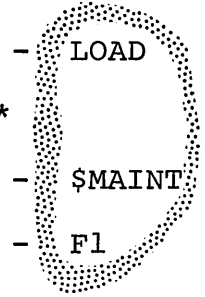
- Library Maintenance program is loaded from the fixed disk on drive 1 (UNIT-F1 in OCL sequence).
- System programs are in the object library on the fixed disk on drive 1 (LIBRARY-O and FROM-F1 in COPY statement).
- The NAME parameter (NAME-SYSTEM) tells the program to copy the system programs.
- The disk that is to contain the copy is the removable disk on drive 1 (TO-R1).

## Printing Library Directories

```

READY
*****
010  LOAD  NAME  -
011          UNIT -
020  DATE
030  SWITCH
040  FILE  NAME  -
*****

```



### OCL LOAD Sequence

Circled areas are operator responses.

Keywords for which no responses are shown are the ones bypassed.

RUN is the response to MODIFY even though the two words do not appear on the same line.

MODIFY



```

ENTER '// ' CONTROL STATEMENT
// COPY FROM-R1 , LIBRARY-ALL , NAME-DIR , TO-PRINT
ENTER '// ' CONTROL STATEMENT
// END

```

Message printed by Library Maintenance program.

Control statement supplied by the operator.

Program prints directories, then asks for another control statement.

END statement, supplied by operator, ends the program.

### Explanation

- Library Maintenance program is loaded from the fixed disk on drive 1 (UNIT-F1 in OCL sequence).
- All library directories and the system directory on the removable disk on drive 1 are printed (COPY statement):
  1. FROM identifies the disk containing the directories.
  2. LIBRARY indicates which directories are to be printed.
  3. NAME and TO indicates that the program is to be printing directories.

## Replacing a Library Entry: Replacement Coming From Another Disk

### Situation

Assume that you have two versions of an object program:

1. New version on the removable disk on drive 1.
2. Old version on the fixed disk on drive 1.

Both versions have the same name (ACCT) and designation (permanent). You want to replace the old version with the new version.

### Statements

```

READY          - LOAD
*****
010  LOAD      NAME      - $MAINT
011                UNIT  - F1
020  DATE
030  SWITCH
040  FILE      NAME      -
*****

```

```

MODIFY        -

```

```

RUN
ENTER '///' CONTROL STATEMENT

```

```

// COPY FROM-R1 , LIBRARY-0 , NAME-ACCT , TO-F1 , RETAIN-R

```

```

ENTER '///' CONTROL STATEMENT

```

```

// END

```

### OCL LOAD Sequence.

Circled areas are operator responses.

Keywords for which no responses are shown are the ones bypassed.

RUN is the response to MODIFY even though the two words do not appear on the same line.

Message printed by Library Maintenance program.

Control statement supplied by operator.

Program replaces library entry, then asks for another control statement.

END statement, supplied by operator, ends the program.

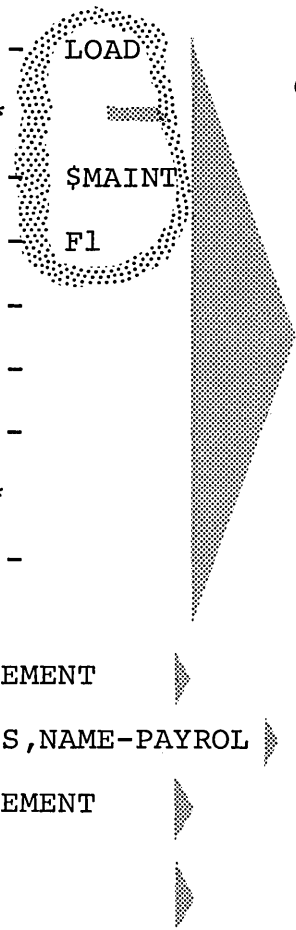
### Explanation

- Library Maintenance program is loaded from the fixed disk on drive 1 (UNIT-F1 in OCL sequence).
- LIBRARY-O, NAME-ACCT, and FROM-R1 in the COPY statement tell the program to read the object program named ACCT from the removable disk on drive 1.
- TO-F1 tells the program to copy the object program to the fixed disk on drive 1. There is no NEWNAME parameter in the COPY statement. Therefore, the name the program will have on the fixed disk is ACCT (NAME-ACCT). Since the old version of the program already exists on the fixed disk 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.

## DELETE EXAMPLES

### Deleting a Temporary Entry From a Library

```
READY - LOAD
*****
010 LOAD NAME - $MAINT
011 UNIT - F1
020 DATE -
030 SWITCH -
040 FILE NAME -
*****
MODIFY -
RUN
ENTER '// ' CONTROL STATEMENT
// DELETE FROM-R1 , LIBRARY-S , NAME-PAYROL
ENTER '// ' CONTROL STATEMENT
// END
```



#### OCL LOAD Sequence.

Circled areas are operator responses.

Keywords for which no responses are shown are the ones bypassed.

RUN is the response to MODIFY even though the two words do not appear on the same line.

Message printed by Library Maintenance program.

Control statement supplied by operator.

Program deletes library entry, then asks for another control statement.

END statement, supplied by operator, ends the program.

#### Explanation

- Library Maintenance program is loaded from the fixed disk on drive 1 (UNIT-F1 in OCL sequence).
- The program deletes a set of source statements (LIBRARY-S in DELETE statement) named PAYROL (NAME-PAYROL) from the removable disk on drive 1 (FROM-R1).
- The absence of a RETAIN parameter implies that the entry designation is temporary. If the designation were permanent, RETAIN-P would have been required.

## Deleting All Temporary Entries With Names That Begin With Certain Characters

READY	-	LOAD		
*****				
010	LOAD	NAME	-	\$MAINT
011		UNIT	-	F1
020	DATE		-	
030	SWITCH		-	
040	FILE	NAME	-	
*****				
MODIFY				
			-	
	ENTER	'//'	CONTROL	STATEMENT
	//	DELETE	FROM-R1,LIBRARY-ALL,NAME-INV.ALL	
	ENTER	'//'	CONTROL	STATEMENT
	//	END		

**OCL LOAD Sequence.**

Circled areas are operator responses.

Keywords for which no responses are shown are the ones bypassed.

RUN is the response to MODIFY even though the two words do not appear on the same line.

Message printed by Library Maintenance program.

Control statement supplied by operator.

Program deletes entries, then asks for another control statement.

END statement, supplied by operator, ends the program.

### Explanation

- Library Maintenance program is loaded from the fixed disk on drive 1 (UNIT-F1 in OCL sequence).
- The entries being deleted are on the removable disk 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).
- The absence of a RETAIN parameter implies that temporary entries are being deleted.

## Deleting All Permanent Library Entries of One Type

```

READY          - LOAD
*****
010   LOAD     NAME     - $MAINT
011           UNIT     -  F1
020   DATE
030   SWITCH
040   FILE     NAME     -
*****

```

### OCL LOAD Sequence.

Circled areas are operator responses.

Keywords for which no responses are shown are the ones bypassed.

RUN is the response to MODIFY even though the two words do not appear on the same line.

```

MODIFY          -
RUN
ENTER '// ' CONTROL STATEMENT ▶
// DELETE FROM-R1 ,LIBRARY-P ,NAME-ALL ,RETAIN-P ▶
ENTER '// ' CONTROL STATEMENT ▶
// END ▶

```

Message printed by Library Maintenance program.

Control statement supplied by operator.

Program deletes entries, then asks for another control statement.

END statement, supplied by operator, ends the program.

### Explanation

- Library Maintenance program is loaded from the fixed disk on drive 1 (UNIT-F1 in OCL sequence).
- The entries being deleted are on the removable disk on drive 1 (FROM-R1 in DELETE statement).
- All permanent procedures are being deleted from the source library (LIBRARY-P,NAME-ALL RETAIN-P).



# RENAME EXAMPLE

## Renaming a Set of Source Statements in a Source Library

```

READY          - LOAD
*****
010   LOAD     NAME      - $MAINT
011           UNIT      -  F1
030   SWITCH
040   FILE     NAME      -
*****
MODIFY        -
RUN
  
```

### OCL LOAD Sequence.

Circled areas are operator responses.

Keywords for which no responses are shown are the ones bypassed.

RUN if the response to MODIFY even though the two words do not appear on the same line.

```

ENTER '//' CONTROL STATEMENT
// RENAME FROM-R1, LIBRARY-S, NAME-ACCT, NEWNAME-ACCT1
ENTER '//' CONTROL STATEMENT
// END
  
```

Message printed by Library Maintenance program.

Control statement supplied by operator.

Program renames entry, then asks for another control statement.

END statement, supplied by operator, ends the program.

### Explanation

- Library Maintenance program is loaded from the fixed disk on drive 1 (UNIT-F1 in OCL sequence).
- The removable disk 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).



## MULTI-VOLUME FILES

### File Statements for Multi-Volume Files

If a file is too large for one disk, you can continue it on one or more subsequent disks. Such files are called multi-volume files. (A volume is one disk.) Multi-volume files can be online or offline. A file is online if all volumes are mounted when the job begins. The UNIT and PACK parameters are equal. An offline file has fewer UNIT parameters (shares same unit).

### Creation

The ways that you can create a multi-volume file depend on the type of file you are creating. For a consecutive and indexed file, the records are stored in consecutive locations on disk, in the order that they are read. One disk is filled at a time.

For consecutive files, each volume must be filled before the next volume is loaded. For indexed files, each volume need not be filled. Each indexed volume is loaded until a keyfield is reached that is higher than the HIKEY for that volume, then the next volume is loaded. Indexed files must be loaded in keyfield sequence. A halt occurs if a volume is filled and there is not a record with a keyfield equal to the HIKEY for that volume. For example, suppose the HIKEY for a volume is 199. You load a record with the keyfield 195. It is less than the HIKEY, so it is loaded on the volume. Next, you load a record with the keyfield 200. Record 200 would be loaded on the next volume, and a halt would occur. The reason for the halt is that you did not load a keyfield record equal to 199 before you jumped to a new volume. This halt can be ignored. You can load the next volume and at some future time insert a keyfield record equal to the HIKEY. To insert a record after the loading sequence has passed, a random add must be done.

Indexed and consecutive files may be either online or offline.

If using removable disks when creating consecutive or indexed files you can mount a disk, wait until the system indicates it is filled. Then, mount the next disk. If you have two drives, you can mount the two disks, wait until the first one is filled, then replace it with the third while your program fills the second disk. In either case, you cannot use more than 52 disks per job.

Space can be allocated on all volumes of a multi-volume file if the volumes are online at the time of the allocation. Space can also be allocated for an offline file, other than the initial volume, but the packs must be empty packs or space (TRACKS and LOCATION) known to be available. You can use both fixed and removable disks with any online multi-volume file.

Direct files must be online. Direct files are created in a non-consecutive manner. When creating such files, you are required to mount all the disks on your disk unit at the same time. The maximum number of disks you could use, therefore, is two if you have only one drive, or three or four if you have two drives.

### Processing

The ways in which you can process multi-volume files depend on the method your program uses to get records from the file. If records are read from a consecutive or indexed file, you can mount a disk, wait until all of the records have been read from the disk, then mount the next disk. If you have two drives, you can mount two disks, wait until all of the records have been read from the first disk, then replace that disk with the third while your program reads from the second disk. When you are processing files offline the disks must be removable. When online, any combination of fixed and removable disks is acceptable, but all must be mounted and must remain mounted.

## OCL Considerations

Multi-volume files, like other disk files, must be described in FILE statements. However, because a multi-volume file involves more than one disk, some FILE keywords require a list of data or codes to describe all of the disks containing the files. This section explains the considerations for using these lists. Each list must begin and end with apostrophes.

## List Requirements

The PACK parameter requires a list. The UNIT parameter may require a list while LOCATION, TRACKS, HIKEY, and RECORDS require a list if they are stated. The considerations for using the lists in these parameters are included in the keyword discussions following.

**KEY LENGTH:** This keyword will be prompted if the response to FILE NAME indicated a multi-volume file (see *Enter Minus* under *End-of-Statement Keys* in Part I). If this is an indexed file, you must respond to KEY LENGTH with a two-digit number 01 through 29. If this is not an indexed file pressing the PROG START key will skip the HIKEY keyword.

**HIKEY:** This keyword must be answered for indexed files. The highest keyfield for each volume must be entered. All characters except commas are allowed as keys. The length of each HIKEY must equal the response to KEY LENGTH and a HIKEY must be entered for each volume. If a HIKEY with fewer characters is entered, blanks will be put into the remaining positions. If an apostrophe is used as part of a HIKEY, it must be entered as two apostrophes or it will be decoded at the end of HIKEY list and an error will occur. When using only one volume of an indexed multi-volume file, the HIKEY must be entered with beginning and ending apostrophes.

The keys in an indexed file can be packed numeric characters. To indicate that a file has packed keys, the operator responds to KEY LENGTH with nn,P where nn is 01-08. Only numeric characters (0-9) are allowed in packed HIKEYS. When responding to HIKEY, the number of characters entered per key is equal to  $2nn-1$ . If the KEY LENGTH response is 07, the HIKEYS would be 13 characters long.

**UNIT:** The keyword UNIT must be followed by a code or codes indicating where the disks that contain the file will be located on the disk unit. No UNIT parameter may be repeated. The codes are as follows:

Code	Meaning
R1	Removable disk on drive one.
F1	Fixed disk on drive one.
R2	Removable disk on drive two.
F2	Fixed disk on drive two.

The order of codes in the UNIT parameter must correspond to the order of names in the PACK parameter.

When you are creating or processing a consecutive or indexed file, you can use the same drive for more than one of the disks; however, the units must then all be removable units. If they are, you must not repeat the code for the drive in the UNIT parameter. When the number of codes in the UNIT parameter is less than the number of names in the PACK parameter, the system uses the codes alternately.

If F1 or F2 is specified, the file must be online multi-volume.

**PACK:** The names of the disks that contain, or will contain, the multi-volume file must follow the keyword PACK. (PACK names must be unique for proper functioning.)

When a multi-volume file is created, the system writes a sequence number on the disks to indicate the order of the disks. The disks are numbered in the order in which you list their names in the PACK parameter.

When a multi-volume file is processed, the system provides two checks to ensure that the disks are used in the proper order.

1. It checks to ensure that the disks are used in the order that their names are listed in the PACK parameter.
2. It checks the sequence numbers of the disks used to ensure that they are consecutive and in ascending order (01, 02, and so on).

The system stops when it detects a disk that is out of sequence. The operator can do one of three things:

1. Mount the proper disk and restart the system.
2. Restart the system and process the disk that is mounted if the sequence is ascending (for consecutive input and update).
3. End the program.

- Consecutive input or update sequence numbers are ignored if the file was not created as multi-volume. If the file is multi-volume and the sequence is ascending but not consecutive, a diagnostic halt is given which allows the proceed option.

**TRACKS or RECORDS:** The keyword **TRACKS** or **RECORDS** must be followed by numbers that indicate the amount of space needed on each of the disks that will contain the multi-volume file. **TRACKS** or **RECORDS** must be specified. Any multi-volume file load requires a **TRACKS** or **RECORDS** keyword whether the file previously existed or not. The order of these numbers must correspond to the order of the names in the **PACK** parameter.

**LOCATION:** The keyword **LOCATION** must be followed by the numbers of the tracks on which the file is to begin on each of the disks you use for the file. The order of the numbers must correspond to the order of the names in the **PACK** parameter. If you omit the **LOCATION** parameter, the system chooses the beginning track on each of the disks. If **LOCATION** is specified for one disk, it must be specified for all disks. If the multi-volume file exists, **LOCATION** must be given and must be identical to the **LOCATION** parameter specified when the file was created.

**RETAIN:** **RETAIN-S** must not be specified unless the file is online multi-volume. If **RETAIN-S** is used for online multi-volume, it cannot be changed to **RETAIN-T** unless also done online.

# OCL CONSIDERATIONS FOR MULTI-VOLUME FILES

KEYWORDS		SEQUENTIAL FILES		DIRECT FILES
		Indexed	Consecutive	
Maximum Number of Disks		10 disks per file statement, 52 disks per job.		Single Drive—2 disks Two Drives—4 disks
UNIT	Location Requirements	R1 or R2 for offline files No restriction for online files		No restriction
	Restrictions on Disk	At file creation time only: <ul style="list-style-type: none"> <li>● First disk can also contain programs, procedures, other files.</li> <li>● Remaining disks must be used only for the one file.</li> </ul>		All the disks used for the file can also contain programs, procedures, other files.
	Operating Considerations	Single Drive Disks must be mounted one at a time Two Drives Disks must be mounted in sequence specified in UNIT statement.		All disks must be on-line during processing.
	Relation to	One entry in the UNIT statement can correspond to more than one disk name in the PACK statement.		A one-to-one correspondence is required between the entries in the UNIT statement and the disk names in the PACK statement.
PACK		When processing a file (or a subset of a file) the disk names must be in the same sequence as they were at file creation time.		
KEY LENGTH		Length must be less than 30 (01—08 if packed keys).	Not used: pressing PROG START will also bypass HIKEY prompt.	
HIKEY		HIKEY responses must correspond one-for-one with the disk names in the PACK statement.		

**KEYWORDS**

**SEQUENTIAL FILES**

**DIRECT FILES**

TRACK  
-or-  
RECORDS



At file creation time:

- Number of tracks (or records) must be specified for each disk.
- Number in TRACKS (or RECORDS) statement must correspond one-for-one with the disk names in the PACK statement.

During subsequent runs:  
TRACKS (or RECORDS) statement can be included in the OCL sequence.  
(For greater detail see keyword descriptions of TRACKS/RECORDS.)

LOCATION



- If specified:  
Addresses must correspond, one-for-one with disk names in PACK statement
- If not specified:  
System will allocate space on each disk.

## CODING MULTI-VOLUME FILE STATEMENTS

1. The operator must begin and end each statement with an apostrophe.
2. The system displays information about each volume on a separate line.
3. The system assigns one statement number to the entire file statement.

## CHANGING MULTI-VOLUME FILE STATEMENTS WITH MODIFY KEYWORD

When using MODIFY keyword to change a multi-volume file statement (other than HIKEY), the entire response to the keyword must be re-entered on one line, separated by commas, with beginning and ending apostrophes.

### Example

	UNIT Statement is	Should be
041	UNIT - 'F1 - R1 - R2 - F2'	UNIT -'F1 -R1 -F2 -R2'

To change at MODIFY time

```
MODIFY
041      - 'F1,R1,F2,R2'
RUN
```

## SAMPLE JOB 7. UPDATING MULTI-VOLUME MASTER FILE

Every Monday the XYZ Novelty Company prepares customer invoices, updates their customer master file, and updates their inventory file. Because the company has a huge customer file they've had to put the file on two disks: customer names beginning with A-L on one disk and the remaining customer names on a second disk. When he created this multi-volume master file, XYZ's programmer assigned the following identifying information:

1. A-L customer names:  
FILE NAME - CMASTER  
PACK - VOL01
2. M-Z customer names:  
FILE NAME - CMASTER  
PACK - VOL02

Because the company often needs information on individual customers, the programmer designed the customer master file as a direct file. The program to update the customer master file is CMUPDA. Here are the OCL statements for the job.

```
READY--                                LOAD (P/S)
*****
010 LOAD                               CMUPDA (P/S)
011                                     UNIT-- F1 (P/S)
020 DATE (12/08/70) -                  (P/S)
030 SWITCH (00000000) -                 (P/S)
040 FILE                               CMASTER (P/S)
041                                     UNIT-- 'F1 (P/S)
                                           - R1' (P/S)
042                                     PACK-- 'VOL01 (P/S)
                                           - VOL02' (P/S)
043                                     LABEL-- (ENTER--)
050 FILE                               (P/S)
*****
MODIFY
RUN (P/S)
```



## Explanation

- 041 UNIT — 'F1  
R1'  
The single quotation marks tell the system the file CMASTER is a multi-volume file. F1, R1 tells the system the file is split between the fixed and removable disks on drive one.
  
- 042 PACK — 'VOL01  
VOL02'  
The single quotation marks tell the system the file is on more than one disk pack. VOL01, VOL02 tells the system the name of the disk packs containing the file. Pressing the ENTER-key causes the system to bypass the rest of the file keywords and prompt FILE NAME.
  
- 050 FILE NAME — Pressing the PROG START key causes the system to bypass all the file keywords and prompt MODIFY.

## SAMPLE JOB 8. CREATING A MULTI-VOLUME INDEXED FILE

We are creating an inventory file. The file is very large and requires five packs. It is an indexed file with a 15 position keyfield; the keyfield consists of part number and warehouse location. The file is divided among the five volumes as follows:

Volume	I01	Keyfields	000-000-000W1B1	to
			175-200-233W1B2	
	I02		175-200-233W1B3	to
			380-456-280W3R6	
	I03		380-456-287W7B3	to
			629-384-300W3F6	
	I04		629-384-301W7B6	to
			949-475-849W8F8	
	I05		949-476-836W4F8	to
			999-999-999W9F9	

The processing starts with I01 on unit R1 and I02 on unit R2. After processing I01, the program processes I02 allowing the operator to remove I01 and mount I03 on unit R1. Likewise, I04 replaces I02 and I05 replaces I03.

```

READY-                                LOAD (P/S)
*****
010 LOAD          NAME-                CRTINV (P/S)
011              UNIT-                F1 (P/S)
020 DATE    (12/31/23) -              (P/S)
030 SWITCH (00000000) -              (P/S)
040 FILE          NAME-                INVMSTR (ENTER-)
              KEY LENGTH-            15 (P/S)
04A              HIKEY-              '175-200-233W1B2 (P/S)
04B              HIKEY-              380-456-280W3R6 (P/S)
04C              HIKEY-              629-384-300W3F6 (P/S)
04D              HIKEY-              949-475-849W8F8 (P/S)
04E              HIKEY-              999-999-999W9F9' (P/S)
041              UNIT-              'R1 (P/S)
              -                      R2' (P/S)
042              PACK-              'VOLI01 (P/S)
              -                      VOLI02 (P/S)
              -                      VOLI03 (P/S)
              -                      VOLI04 (P/S)
              -                      VOLI05' (P/S)
043              LABEL-              (P/S)
044              RECORDS-            (P/S)
045              TRACKS-            '100 (P/S)
              -                      193 (P/S)
              -                      150 (P/S)
              -                      193 (P/S)
              -                      80' (P/S)
046              LOCATION-          '87 (P/S)
              -                      8 (P/S)
              -                      49 (P/S)
              -                      8 (P/S)
              -                      8' (P/S)
047              RETAIN-            P (ENTER-)
050 FILE          NAME-              (P/S)
*****
MODIFY

RUN (P/S)

```

**Explanation**

- **KEY LENGTH:** All characters except commas are allowed as part of the HIKEY. If apostrophes are used as part of the key, two apostrophes must be entered for each one in the key. The number of characters entered for HIKEYs must equal KEY LENGTH.
- **045 TRACKS**      The file need not occupy the entire volume if the number of tracks and the starting location are given. You must be sure these areas are available because the system cannot check offline packs.
- **046 LOCATION**

No statement number is assigned KEY LENGTH. This keyword cannot be changed at MODIFY time.

**SAMPLE JOB 9. MAINTAINING A MULTI-VOLUME INDEXED FILE WITH PACKED KEYS**

We are maintaining a multi-volume indexed file. The file occupies four volumes. The keyfield is 15 characters long in packed format. The keyfield takes eight bytes in the record. The file is divided as follows:

```

Volume P01  Keyfields  000 000 000 000 000
                through
                000 025 000 000 000
           P02          000 025 000 000 001
                through
                000 050 000 000 000
           P03          000 050 000 000 001
                through
                000 075 000 000 000
           P04          000 075 000 000 001
                through
                000 100 000 000 000
    
```

The OCL required to use this file is as follows:

```

READY-                                LOAD (P/S)
*****
010 LOAD                               NAME-   PAYROL (P/S)
011                                UNIT-   F1 (P/S)
020 DATE (07/09/71) -                (P/S)
030 SWITCH (00000000) -              (P/S)
040 FILE                               NAME-   PAYROLL (ENTER -)
                                KEY LENGTH- 08,P (P/S)
04A                                HIKEY-   '0000250000000000 (P/S)
04B                                HIKEY-   0000500000000000 (P/S)
04C                                HIKEY-   0000750000000000 (P/S)
04D                                HIKEY-   0001000000000000' (P/S)
041                                UNIT-   'R1 (P/S)
                                -           R2' (P/S)
042                                PACK-   'VOLP01 (P/S)
                                -           VOLP02 (P/S)
                                -           VOLP03 (P/S)
                                -           VOLP04' (P/S)
043                                LABEL-   ACCONT (ENTER -)
050 FILE                               NAME-   (P/S)
*****
MODIFY
RUN (P/S)
    
```

## INCLUDING SORT SOURCE OR UTILITY CONTROL STATEMENTS IN A PROCEDURE

The INCLUDE option can be used during MODIFY time of a BUILD cycle to include sort source or utility control statements in a procedure. This is useful if the control statements are long or complex and the job is run frequently. A maximum of 25 control statements can be included in each procedure.

During the BUILD cycle, the INCLUDE option must be the last MODIFY option used. After the included statements are keyed in, the RUN entry then puts the procedure and included statements in the source library.

The CALL cycle will be different if the called procedure has included statements. After the

OCL statements are printed, MODIFY will be prompted to allow changes to the OCL statements. After the operator types RUN, the system will print INCLUDED STATEMENTS and then list the statements. MODIFY will now be prompted again, to allow changes to be made to the included statements. The operator types RUN to run the job.

For an example of Including Sort Source Statements in a procedure see the *IBM System/3 Disk Sort Reference Manual, SC21-7522*.

An example of including Utility Control statements in a procedure follows.

## SAMPLE JOB 10. INCLUDING UTILITY CONTROL STATEMENTS IN A PROCEDURE

Sample job 1 showed an OCL LOAD cycle for initializing the removable disk on drive one. This sample job shows how to do the same job using BUILD and CALL cycles and including the Utility Control Statements in the procedure.

```

READY-
000 BUILD          NAME-          BUILD (P/S)
001                UNIT-          INITRI (P/S)
*****
010 LOAD          NAME-          $INIT (P/S)
011                UNIT-          F1 (P/S)
020 DATE          -              (P/S)
030 SWITCH (00000000) -          (P/S)
040 FILE          NAME-          (P/S)
*****
MODIFY

INCLUDE (P/S)
*****
ENTER UTILITY CONTROL STATEMENTS
00

// UIN UNIT=R1,TYPE=PRIMARY (P/S)
01

// VOL PACK=12345 (P/S)
02

// END (P/S)
03

RUN (P/S)
*****
MODIFY

```

```

READY-                                CALL (P/S)
000 CALL                             NAME-    INITRI (P/S)
001                                  UNIT-    F1 (P/S)
*****
010 LOAD                             NAME-$INIT
011                                  UNIT-F1
*****
MODIFY

RUN (P/S)
*****
INCLUDED STATEMENTS
00 // UIN UNIT-R1,TYPE-PRIMARY
01 // VOL PACK-12345
02 // END
*****
MODIFY

RUN (P/S)

```

### INCREASING FILE SIZE OF THE RPG PROCEDURE

The IBM-supplied compile procedure can only compile RPG II programs with less than 400 statements. To compile larger programs, the file statements must be modified to increase their size above 10 tracks (see *Modify; Changing a Previous OCL Statement* in Part I). Using the MODIFY option will only increase the file size for one compile. The RPG II procedure will not be changed in the source library. To change the procedure in the source library you must either build a new procedure (see *BUILD NAME* in Part I) or use the KSE utility program.

### ENTERING RPG II SOURCE STATEMENTS FROM THE KEYBOARD AT COMPILE TIME

The IBM-supplied compile procedure requires that the RPG II source statements be in the source library of a disk. By using the Keyboard Source Entry Utility (\$KSE), source statements can be format checked and syntax checked as they are put on disk.

The source statements can, however, be entered from the keyboard at compile time. These statements are read by the compiler and checked for format errors. If any errors are found they cannot be corrected and the compile will not be successful. The compile job must be rerun and all source statements keyed in again.

To key in source statements from the keyboard, the IBM-supplied compile procedure RPG is used. This procedure does not prompt COMPILE OBJECT, SOURCE, or UNIT.

## CHAINED PROCEDURES

A finished job usually requires that more than one program be run. Several customer programs with utility programs between them may be required to complete the finished report. This sequence of programs can be put in chained procedures.

By chaining procedures, several benefits can be realized, including:

- Programs are always run in the correct sequence.
- Operator intervention and, therefore, chance of operator error, is decreased.
- File space can be saved. Files used to pass data from job to job can be scratched after the last program.
- Files are less likely to be destroyed by running non-related programs between programs of a job.

To chain procedures, the operator first builds a master procedure to chain together other procedures. This is done by responding to READY with BUILDDC. The system will then repetitively prompt CALL NAME and UNIT, allowing the operator to respond with the name and unit of the procedures that are to be chained. When all procedure names have been entered, the operator responds to CALL NAME or UNIT with the ENTER MINUS (ENTER-) key. The system then allows the operator to MODIFY the entries. When RUN is entered, the master procedure is put in the source library as a permanent entry.

Master procedures can call other master procedures up to 9 levels. The original master procedure called (level 1) can call another master procedure (level 2), which can call another master procedure (level 3),

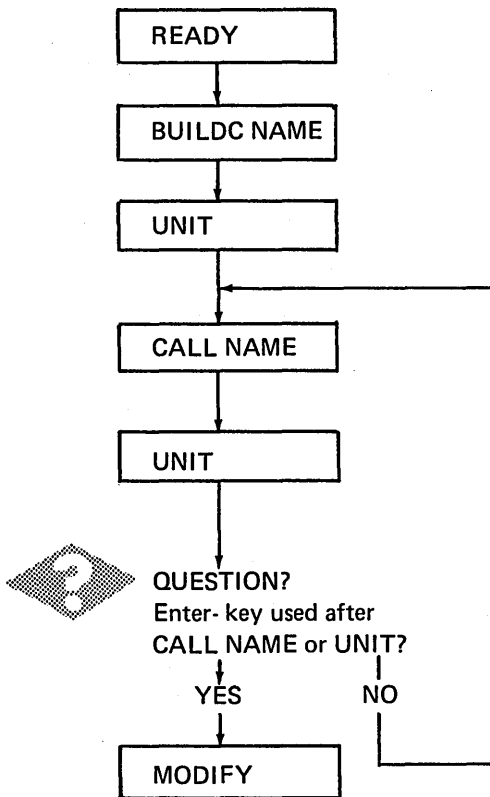
etc., on up to 9 levels. Care must be taken to avoid calling a master procedure that was already called earlier in the chain or an endless loop will result. A master procedure can contain only CALL and UNIT statements.

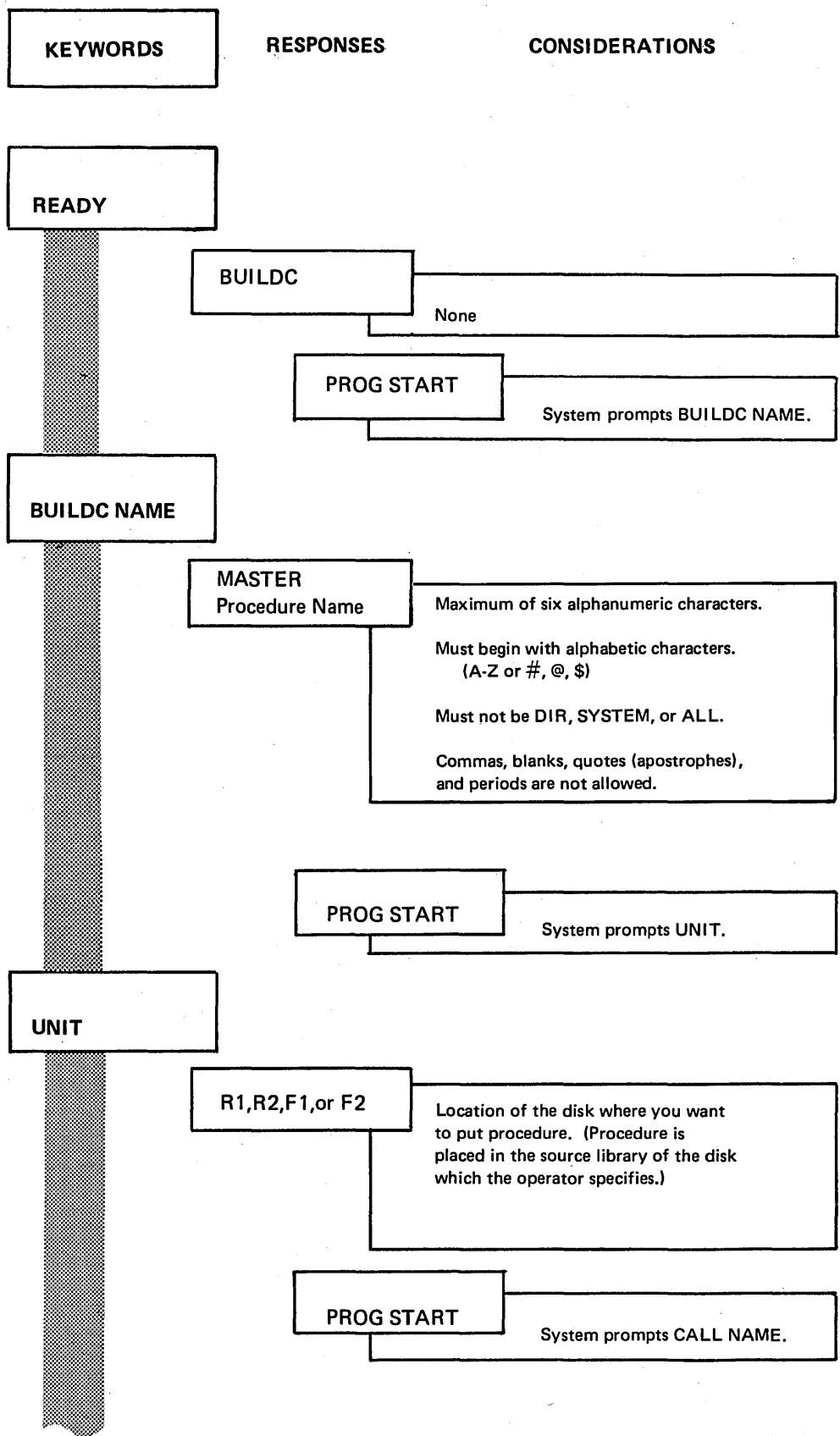
Delayed responses are not allowed in a BUILDDC cycle. However, the called procedures can contain delayed responses.

To run the chained procedures, the operator initiates a CALL cycle and responds to CALL NAME with the name of the master procedure. Each procedure is then called by the master procedure and run.

When running chained procedures, the operator is never prompted MODIFY to make changes.

If HALT is specified, the system will not halt until the last job of a chain is complete.





**KEYWORDS**

**RESPONSES**

**CONSIDERATIONS**

**CALL NAME**

Name of Procedure

Name of a procedure in the source library. The procedure can be an IBM-supplied procedure (RPGB) or a procedure created by a BUILD or BUILD C cycle.

PROG START

System prompts UNIT.

or

ENTER—

System prompts UNIT then MODIFY.

**UNIT**

R1,R2,F1, or F2

Location of the disk whose source library contains the procedure.

PROG START

System prompts CALL NAME (or MODIFY if ENTER — used after CALL NAME).

or

ENTER—

System prompts MODIFY.



**KEYWORDS**

**RESPONSES**

**CONSIDERATIONS**

**MODIFY**  
(Operator can use one, all, or a combination of the responses.)

<b>LOG</b>	Used only if CRT display or 22" printer is on system (see Appendixes D and E).
<b>PROG START</b>	System prompts LOG DEVICE.
<b>CANCEL</b>	Cancel job.
<b>PROG START</b>	System prompts READY or displays end-of-job halt.
<b>FORMS</b>	Change lines per page printed output for system programs.
<b>PROG START</b>	System prompts FORMS DEVICE.
<b>Asterisk (*) Followed by comments</b>	Enter comment.
<b>PROG START</b>	System waits for next MODIFY response.
<b>Statement number and comma</b>	To delete statement.
<b>PROG START</b>	System waits for next MODIFY response.

**KEYWORDS****RESPONSES****CONSIDERATIONS****Statement number**

To correct statement.

**PROG START**

System waits for correct statement.

**RUN**Tells system—  
a. The cycle is complete.  
b. Run the job.**PROG START**

System runs job.

## SAMPLE JOB 11. CHAINED PROCEDURE

We're going to use the BUILD cycle to chain two procedures created with the BUILD cycle. First, we use the BUILD cycle to build procedures to use the Conversational Utilities (\$KSE and \$KDE).

After the chained procedure is built, the CALL cycle is used to run the chained procedures.

```
READY-                                BUILD (P/S)
000 BUILD          NAME-              KSE (P/S)
001                UNIT-             F1 (P/S)
*****
010 LOAD           NAME-              $KSE (P/S)
011                UNIT-             F1 (P/S)
020 DATE           -                  (P/S)
030 SWITCH (00000000) -              (P/S)
040 FILE           NAME-              (P/S)
*****
MODIFY

RUN (P/S)
```

```
READY-                                BUILD (P/S)
000 BUILD          NAME-              KDE (P/S)
001                UNIT-             F1 (P/S)
*****
010 LOAD           NAME-              $KDE (P/S)
011                UNIT-             F1 (P/S)
020 DATE           -                  (P/S)
030 SWITCH (00000000) -              (P/S)
040 FILE           NAME-              KDEFILE (P/S)
041                UNIT-             F1 (P/S)
042                PACK-              F1F1F1 (P/S)
043                LABEL-             DRIV2 (P/S)
044                RECORDS-           4 (P/S)
045                LOCATION-          (P/S)
046                RETAIN-            T (P/S)
047                DATE-              (P/S)
050 FILE           NAME-              (P/S)
*****
MODIFY

RUN (P/S)
```

```

READY-
000 BUILD  NAME-      BUILD (P/S)
001         UNIT-     MASTER (P/S)
001         UNIT-     F1 (P/S)
*****
010 CALL   NAME-      KSE (P/S)
011         UNIT-     F1 (P/S)
020 CALL   NAME-      KDE (P/S)
021         UNIT-     F1 (ENTER-)
*****
MODIFY
RUN (P/S)

```

```

READY-
000 CALL   NAME-      CALL (P/S)
001         UNIT-     MASTER (P/S)
000 CALL   NAME-KSE
001         UNIT-F1
*****
010 LOAD   NAME-$KSE
011         UNIT-F1
*****

```

```

FORMAT DESCRIPTION ?      YES (P/S)
FORMAT TYPE -            KDE (P/S)
NEW SOURCE MODULE ?      YES (P/S)
SOURCE MODULE NAME -     KDEFOR (P/S)
SOURCE MODULE UNIT -     F1 (P/S)
06672 NEW STATEMENTS MAY BE ADDED TO SOURCE ENTRY
00000   H01              096 (P/S)
00010   A005 (P/S)
00020   A091 (P/S)
00030   H02              (COMMAND KEY 06 PRESSED)
END OF JOB ?             YES (P/S)
KSE END OF JOB

```

```

000 CALL          NAME-KDE
001              UNIT-F1
*****
010 LOAD          NAME-SKDE
011              UNIT-F1
020 FILE          NAME-KDEFILE
021              UNIT-F1
022              PACK-F1F1F1
023              LABEL-DRIV2
024              RECORDS-4
025              RETAIN-T
*****

```

FORMAT NAME - KDEFOR (P/S)

FORMAT UNIT - F1 (P/S)

DISPLAY FORMATS ? YES (P/S)

H01096

A005

A091

NEW KDE FILE ? YES (P/S)

KEY FIELD START - NO (P/S)

SELECT FORMAT NUMBER - 01 (P/S)

\* \*

00000 THIS IS AN EXAMPLE OF CHAIN PROCEDURE ON THE MODEL 6 (P/S)

00010 KSE WAS THE FIRST JOB EXECUTED AND KDE WAS THE SECOND AND LAST JOB (P/S)

00020 THE CHAIN WAS INITIATED BY CALLING MASTER, WHICH WAS BUILT IN A BUILD CYCLE (P/S)

00030 (COMMAND KEY 06,PRESSED)

\*\*\*\*\*

BATCH ACCUMULATORS	00	01	02	03	04
	0	0	0	0	0
	05	06	07	08	09
	0	0	0	0	0
FINAL ACCUMULATORS	00	01	02	03	04
	0	0	0	0	0
	05	06	07	08	09
	0	0	0	0	0

\*\*\*\*\*

END OF JOB ? YES (P/S)

KDE END OF JOB

## APPENDIX B: RECORDS – TRACKS CONVERSION

### For Sequential or Direct Files

To determine how many tracks will be required for a sequential or direct file:

1. Number of records x record length = total number of characters.
2. Total number of characters  $\div$  6144 <sup>①</sup> = number of tracks. (Round result up to nearest whole number.)

### For Indexed Files

To determine how many tracks will be required for an indexed file:

#### Step 1. (Tracks Required for Data)

- A. Number of records x record length = total number of characters.
- B. Total number of characters  $\div$  6144 = number of tracks. (Round result up to nearest whole number.)

#### Step 2. (Tracks Required for Index)

- A. Key Field length + 3 = index entry length.
- B.  $256 \div$  <sup>②</sup> index entry length = number of entries per sector. (Round result down to nearest whole number.)
- C. Number of records  $\div$  number of entries per sector = number of sectors. (Round result up to nearest whole number.)
- D. Number of sectors  $\div$  24 <sup>③</sup> = number of tracks. (Round result up to nearest whole number.)

#### Step 3. (Total Track Requirement)

Result of step 1 + result of step 2 = total number of tracks required for the indexed file.

① Number of characters in a track.

② Number of characters in a sector.

③ Number of sectors per track.

Disk Area	Contents
VTOC*	Detailed information about each file on disk
Source Library	Source Library Directory RPG II Source Programs Sort Specifications Procedures KSE Input (Format Descriptions or Source Statements)
Object Library	Object Library Directory Compiled Programs System Programs
Files	User files System files

\*Volume Table of Contents

**Volume Table of Contents (VTOC)**

The VTOC contains detailed information about each file on the disk. Much of this information is for system use only and is of no interest to the programmer. The VTOC file information significant to the programmer is:

1. Name.
2. Starting track location and number of tracks.
3. Designation (Permanent, Temporary, or Scratch).
4. Organization (Sequential, Direct, or Indexed).
5. Creation date.

**Source Library**

Procedures, RPG II source programs, and KSE input always reside in a source library. The source library directory contains the name and address (track and sector) of each procedure, RPG II source program, and set of KSE input in the library.

**Object Library**

Compiled programs and system programs always reside in an object library. The object library directory contains the name and address (track and sector) of each program in the library.

**Files**

Identifying information about every file on a disk is contained in the disk VTOC.

A disk is limited to 50 files because the VTOC has space for identifying only that many files.

## APPENDIX D: OCL FOR THE 22" PRINTER (IBM 2222 PRINTER)

The optional 22" printer provides the MODEL 6 system with the ability to print on two forms. Each form has its own forms tractor. The left tractor is called PRIMARY and the right tractor is SECONDARY.

### Using the FORMS Statement with the 22" Printer

The lines per page setting of the PRIMARY and SECONDARY tractors can be different. (For example, the PRIMARY tractor could print 25 lines per page, while SECONDARY

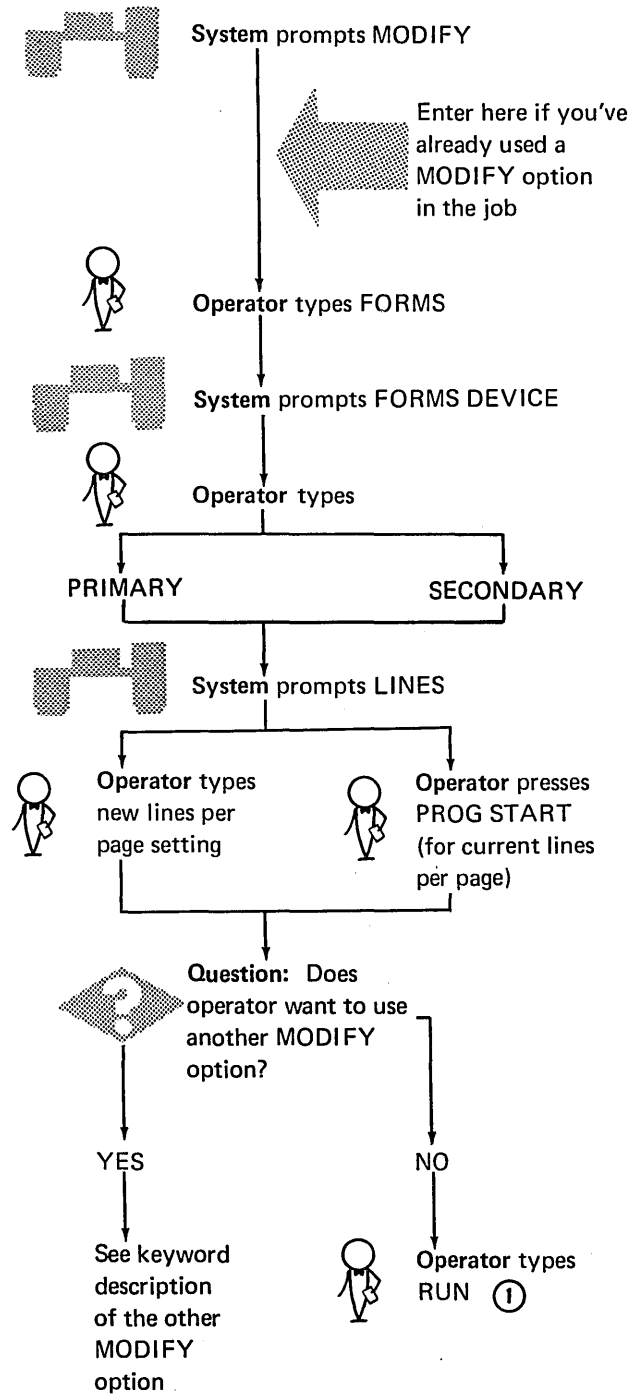
prints the standard 66 lines per page.) Separate settings are specified by entering different FORMS statements for each tractor during the MODIFY phase.

### Log Device

The log device is used to print OCL statements and messages. The PRIMARY tractor will be the log device at IPL time when the 2222 Printer is used. The secondary tractor can be assigned as the logging device by entering LOG at either READY or MODIFY time. If the secondary tractor is the logging device, logged data begins in print position 110. (See *READY-Entering LOG* and *MODIFY-Entering LOG* in Appendix E.)



## MODIFY – Entering the Keyword FORMS



- ① Whenever the keyword FORMS is entered in an OCL sequence a system halt occurs after RUN in case the operator needs to change the paper in the printer. The system remains idle until the operator presses the PROG START key.

## APPENDIX E: OCL FOR THE IBM 2265-2 DISPLAY

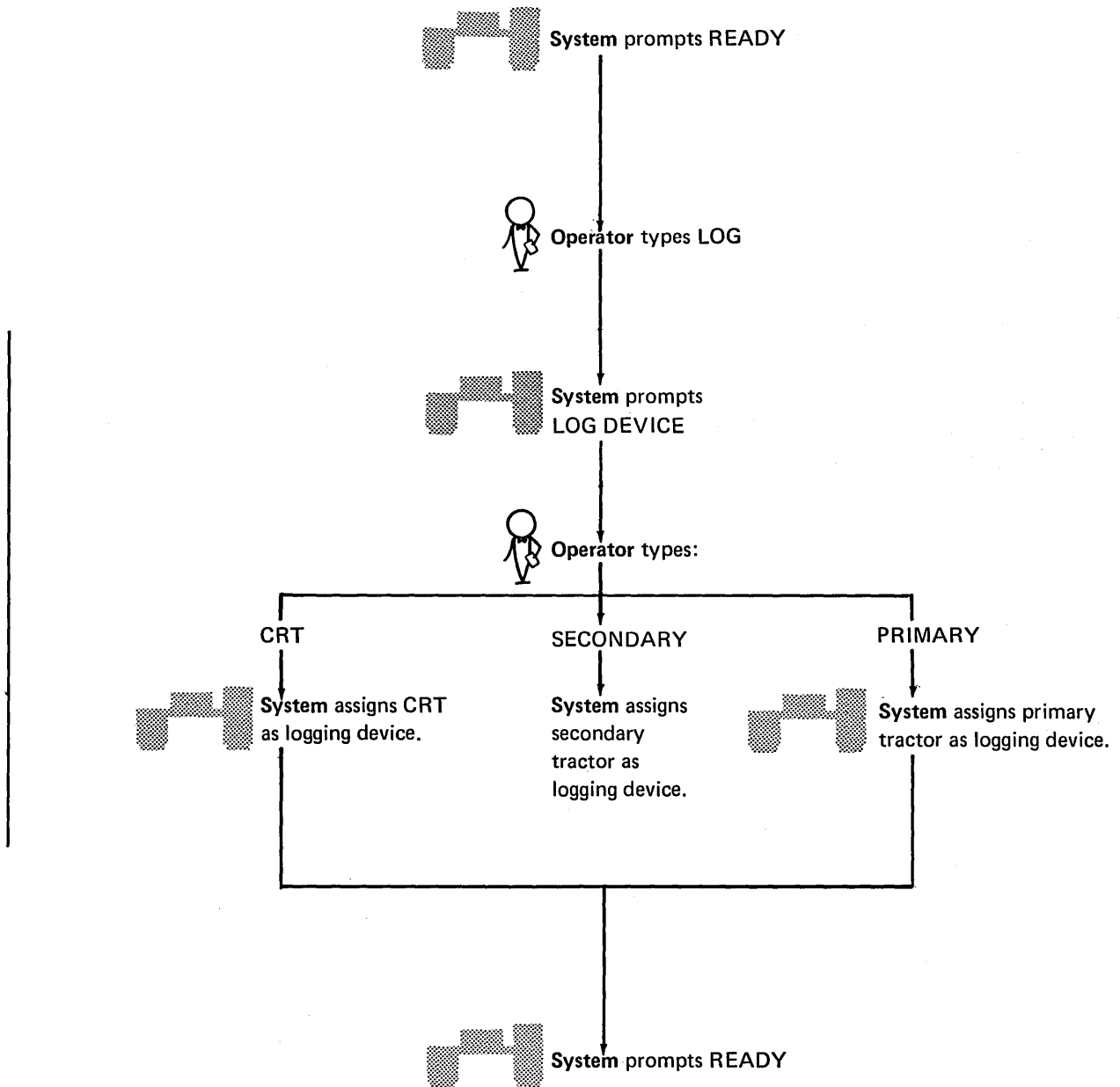
The IBM 2265-2 display unit can be used as the system logging device. The logging device displays conversational OCL statements, utility control statements, job comments, and error messages and codes. The log device can also be used for normal output from the job being run.

When the 2265-2 (CRT) is used as the logging device, an additional 1K of core storage is needed for the system, thus reducing the core available for the user program. This extra core is not needed if the user program specifies the CRT as an output device.

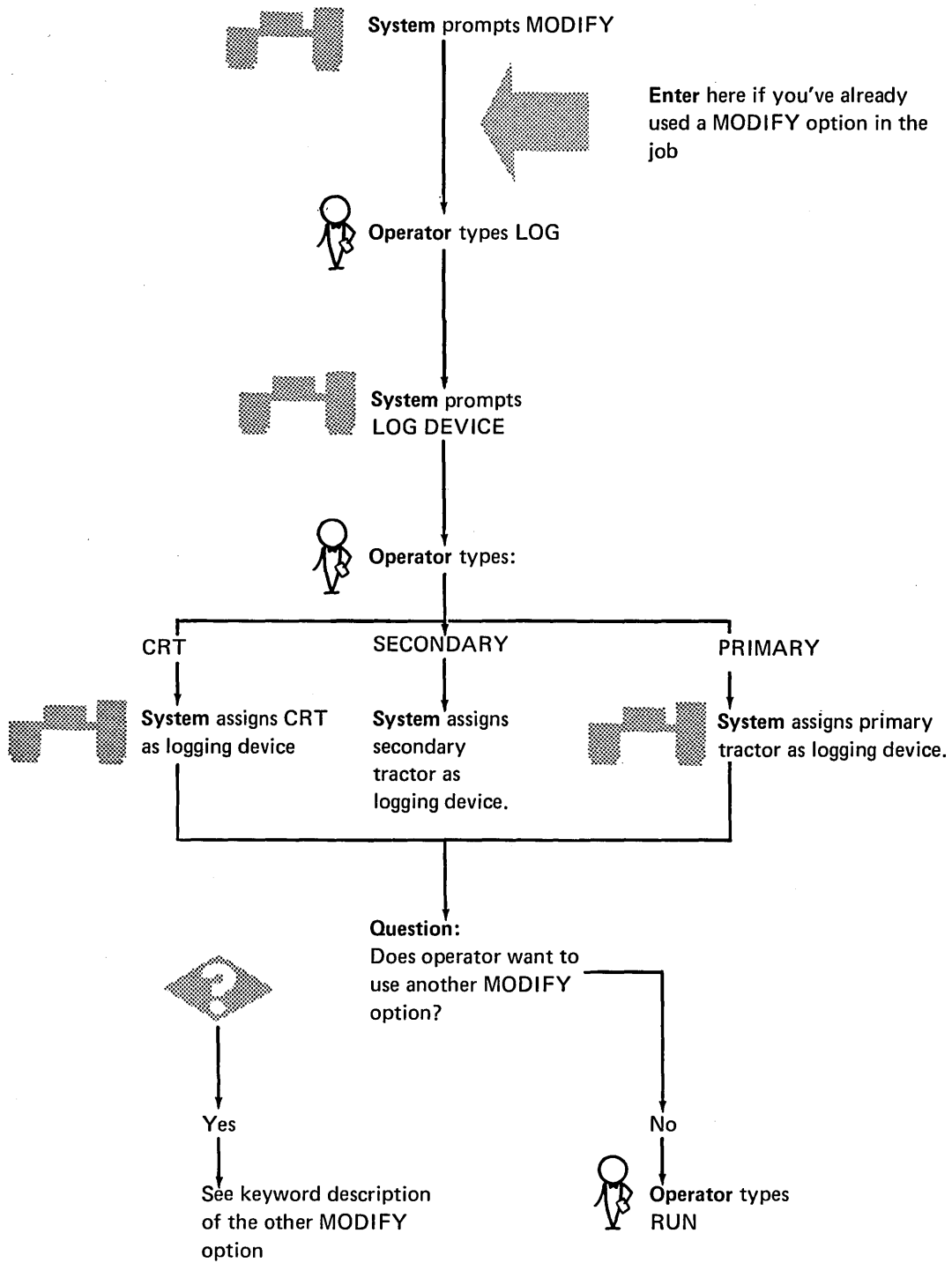
The operator can assign either the CRT display or the printer as the logging device. If the operator changes the logging device the change remains in effect until either:

- The operator specifically overrides the change with another LOG statement.
- The next IPL procedure.

# READY – Entering LOG



# MODIFY—Entering LOG



# APPENDIX F: OPERATOR'S OCL GUIDE

The operator's OCL guide will be available for you to use to tell your operator how to respond to the OCL prompting for a job. The CALL cycle is not included on the guide because the OCL prompting for that cycle is so short.



International Business Machines Corporation  
System/3 Model 6

GX21-9126  
Printed in U.S.A.

Job \_\_\_\_\_

Date \_\_\_\_\_

## OPERATION CONTROL LANGUAGE (OCL) GUIDE

Programmer \_\_\_\_\_

Keywords										Responses										Considerations											
R	E	A	D	Y						B	U	I	L	D																Procedure Name	
0	0	0			B	U	I	L	D																					F1, R1, F2 or R2	
0	0	1																												Columns 75-80 of RPG Control Card or System Program Name	
0	1	0			L	O	A	D																						F1, R1, F2 or R2	
0	1	1																												F1, R1, F2 or R2	
0	2	0			D	A	T	E																						mmddyy or ddmmyy	
0	3	0			S	W	I	T	C	H																					1-On, 0-Off, X-No Change
0	4	0			F	I	L	E																						Columns 7-14 of RPG File Description Specifications or Predefined File Name	
0	4	1																												F1, R1, F2 or R2	
0	4	2																												Disk Name (Assigned by Disk Initialization Program)	
0	4	3																												VTOC File Name (if different than response to FILE NAME)	
0	4	4																												1-999999 (Maximum Number of Records in File)	
0	4	5																												1-398 (Maximum Number of Tracks for this File)	
0	4	6																												8-405 Location of First Track of File	
0	4	7																												S-Scratch, T-Temporary, P-Permanent	
0	4	8																												mmddyy or ddmmyy	
0	5	0			F	I	L	E																						Columns 7-14 of RPG File Description Specifications or Predefined File Name	
0	5	1																												F1, R1, F2 or R2	
0	5	2																												Disk Name (Assigned by Disk Initialization Program)	
0	5	3																												VTOC File Name (if different than response to FILE NAME)	
0	5	4																												1-999999 (Maximum Number of Records in File)	
0	5	5																												1-398 (Maximum Number of Tracks for this File)	
0	5	6																												8-405 Location of First Track of File	
0	5	7																												S-Scratch, T-Temporary, P-Permanent	
0	5	8																												mmddyy or ddmmyy	
M	O	D	I	F	Y																									<p style="text-align: center;"><u>MODIFY OPTIONS</u></p> <p>1. Enter RUN</p> <p>2. Enter CANCEL</p> <p>3. Correct Statement</p> <p style="padding-left: 20px;">Enter Statement number</p> <p style="padding-left: 20px;">Retype or delete (.) response</p> <p>4. Create new Statement</p> <p style="padding-left: 20px;">INCLUDE, LOG, FORMS, *(For Comments)</p>	

## APPENDIX G: CARD OCL FOR MODEL 6

The IBM 5496 Data Recorder, Model 1, with System/3 Model 6 On-Line Feature provides card input/output capability for System/3 Model 6. The data recorder is selected as system input device during OCL prompting. Control is returned to the keyboard by entering a READER statement in the data recorder or by performing another program load procedure.

### ASSIGNING DATA RECORDER AS SYSTEM INPUT DEVICE

System Prompts    Operator Enters

At IPL time        DATE —        current date  
                    READER —        DATA96

Between jobs      READY —        READER  
                    READER —        DATA96

Following the DATA96 response, all OCL must be entered in card form from the data recorder.

At the time the data recorder is selected as system input device the following switch settings must be:

Operator Console — DATA RCRDR switch to ON LINE  
  
Data Recorder — AUTO REL switch to ON, all others OFF.

### RETURNING CONTROL TO KEYBOARD

The keyboard is reassigned as system input device by doing either of the following:

- Enter a /& statement followed by a // READER KEY statement from the Data Recorder. These statements must be placed after a // RUN statement and before a // LOAD or // CALL statement.
- Perform a program load from the operator console.

### CARD FORMAT OF OCL STATEMENTS

The following OCL statements can be loaded from the data recorder. The parameters of the statements that are prompted in conversational mode are described elsewhere in this book. The statements that are allowed with card input are described in the notes following this list.

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.

Application	
Program Name	Number

OCL STATEMENTS

1    4    8    12    16    20    24    28    32    36    40    44    48    52    56    60    64

- ①
- ②
- ③
- ④
- ⑤
- ⑥
- ⑦
- ⑧

```
// LOAD Program-name,unit
// LOAD *
// CALL Procedure-name,unit
// RUN
// READER KEY
// SWITCH xxxxxxxx
// COMPILE OBJECT-unit, SOURCE-name, UNIT-unit
// FORMS DEVICE-PRIMARY, LINES-number
// FORMS DEVICE-SECONDARY, LINES-number
// LOG ON
// LOG OFF
// LOG CRT
// LOG PRIMARY
// LOG SECONDARY
// FILE NAME-filename, UNIT-unit, PACK-name,
// LABEL-filename, RECORDS-number, TRACKS-number,
// LOCATION-track number, RETAIN-code, DATE-date
// NOHALT
// HALT
// PAUSE
// *
// &
// *
```

- ① An asterisk indicates that the object program will be loaded from the data recorder. Program-name and unit parameters must not be included. The cards that contain the program must follow the RUN statement for the program and must be followed by /\* to indicate the end of the object deck.
- ② OBJECT-unit must be the first parameter on the statement.
- ③ The DEVICE parameter is optional. The LINES parameter must be present.
- ④ The log device must be on when the system is in conversational mode.
- ⑤ LABEL, RECORDS or TRACKS, LOCATION, RETAIN, and DATE parameters are optional. NAME-filename must be the first parameter on the statement.
- ⑥ During card input, the system halts each time a /\* (end-of-job) or /& statement is read. The NOHALT statement allows the system to start the next job without a halt. The HALT statement is used to cancel a NOHALT condition. If the HALT and NOHALT statements are placed in a procedure they are not displayed when the procedure is prompted.
- ⑦ A PAUSE statement entered from the data recorder causes the system to stop until the operator restarts it. PAUSE statements are usually preceded by comments (\*) instructing the operator to perform some function on the system. If PAUSE statements and comments are placed in a procedure the comments are displayed during prompting but the system does not stop.
- ⑧ /\* indicates end-of-job. /& is used as a delimiter and indicates that a new job is starting. If a 3 option (immediate cancel) was taken at a halt in the preceding job the system looks for the next LOAD or CALL statement in the job stream. The /& statement changes this mode and tells the system to read the next // card regardless of what it is. In this manner a // READER KEY statement would be recognized, returning control to the keyboard.



## GENERAL CODING RULES

The rules for coding the OCL statements in cards are as follows:

1. // in positions 1 and 2.
2. One or more blanks between the // and the word that forms the statement identifier (LOAD, RUN, CALL, etc.).
3. One or more blanks between the statement identifier and the first parameter.
4. If you need more than one parameter, use a comma to separate them. No blanks are allowed in or between parameters. Anything following the first blank is considered a comment.
5. If you are writing keyword parameters (XXX-xxx), place the keyword first and use a hyphen to separate the keyword from the code or data.
6. If the parameter is not a keyword parameter, write the parameters in the order in which they are shown. Keyword parameters can be in any order except in the following cases:

// COMPILE      OBJECT-unit must be the first parameter.

// FILE            NAME-filename must be the first parameter.

7. All OCL statements except FILE must not exceed 96 characters. Because of the large number of parameters possible in a FILE statement, you can continue the FILE statement on additional cards. The rules are:

- Place a comma after the last parameter in every card but the last. The comma followed by a blank indicates the statement is continued.

- Begin each new card with // in positions 1 and 2.
  - Leave one or more blanks between the // and the first parameter.
8. Comments can be placed after the parameters on any OCL statement (except HIKEY parameters. See *Coding Multi-Volume File Parameters* in this appendix). Leave one or more blanks after the last parameter and then list the comment. Complete lines of comments are entered with the \* comment statement. Place an \* in column 1 and start the comments in column 2.

## STATEMENT ORDER

- |            |   |
|------------|---|
| /&         | should be the first statement of a job.   |
| // LOAD    | statement must precede RUN statement in job stream. If LOAD*, the cards that contain the program must follow the RUN statement and be followed by a /* statement. |
| // CALL    | statement must precede RUN statement in job stream.   |
| // RUN     | statement must be last statement within the set of statements required to run a program.  |
| // READER  | statement must precede a LOAD or CALL statement and follow a RUN statement.   |
| // SWITCH  | statement must follow a LOAD or CALL statement and must precede a RUN statement.  |
| // COMPILE | statement must follow a LOAD or CALL statement and must precede a RUN statement.  |
| // FORMS   | can appear anywhere in the job stream.  |

// LOG statement must follow a LOAD or CALL statement and precede a RUN statement.

// FILE statements must follow a LOAD or CALL statement and precede a RUN statement.

// HALT can appear anywhere in the job stream.

// NOHALT can appear anywhere in the job stream.

// PAUSE can appear anywhere in the job stream.

\*comments can appear anywhere in the job stream.

/\* (end-of-job) follows a program deck or data file entered from the Data Recorder.

2. The HIKEY parameter must contain HIKEYs separated by commas. When continuation cards are needed for HIKEY parameters, comments are not allowed on the cards, and the data must start in column four of the continuation card.
3. An apostrophe within a HIKEY must be entered as a double apostrophe or it will be decoded as end of HIKEYs, and an error will occur.
4. When using only one volume of an indexed multi-volume file, the HIKEY parameter must be included with beginning and ending apostrophes. The other file parameters must not have apostrophes.
5. To indicate packed keys, HIKEY-P'xxxx, xxxx, xxxx,' must be coded. All characters in packed HIKEYs must be numeric and all packed HIKEYs must be the same length.

### CODING MULTI-VOLUME FILE PARAMETERS

When coding card OCL file statements for multi-volume files these rules must be followed:

1. Each parameter that requires multiple entries must begin and end with a single quote (') and have the entries separated by commas.

Key length is not a parameter for indexed files when OCL statements are entered on cards. Sample job 2 under *Multi-Volume Files* in Appendix A would have the following four OCL file statements if OCL were on cards:

```
// FILE NAME-INVMSTR,UNIT-'R1,R2',
PACK-'VOLI02,VOLI03,VOLI03,VOLI04

// VOLI05',HIKEY-'175-200-233W1B2,
380-456-280W3R6,629-384-300W3F6

// 949-475-849W8F8,999-999-999W9F9',
TRACKS-'100,193,150,193,80'

// LOCATION-'87,8,49,8,8',RETAIN-P
```

## APPENDIX H: OCL ERROR MESSAGES

These messages will be given if errors are made during conversational OCL. Most messages are self-explanatory and will not need further reference, however, if the operator is in doubt as to the meaning of a message references are given.

Number	Message	Meaning
00	NO PROGRAM NAME GIVEN	Response to LOAD NAME was blank.
01	NO UNIT GIVEN	Response to UNIT was blank.
02	INVALID PROGRAM NAME SPECIFIED	Response to LOAD NAME was invalid. See <i>LOAD NAME</i> in Part I.
03	INVALID UNIT SPECIFIED	Response to UNIT was invalid. See <i>UNIT</i> in Part I.
04	PROGRAM NOT FOUND ON SPECIFIED UNIT	The program specified by response to LOAD NAME was not found in the object library of the unit specified by response to UNIT.
05	NO PROCEDURE NAME GIVEN	Response to CALL NAME or BUILD NAME was blank.
06	SOURCE NOT FOUND ON SPECIFIED UNIT	The source module specified by response to SOURCE was not found in source library of unit specified by UNIT.
07	INVALID PROCEDURE NAME	Response to BUILD NAME or CALL NAME was invalid. See <i>BUILD NAME</i> in Part I.
08	MULTI-VOLUME FILE RESPONSES NOT IN 1-1 RATIO	The number of responses to file keywords PACK, HIKEY, LOCATION, TRACKS or RECORDS were not equal.
09	PROCEDURE NOT FOUND ON SPECIFIED UNIT	Procedure specified by response to CALL NAME was not found in source library of unit specified by UNIT.
10	INVALID SWITCH SETTINGS	Response to SWITCH was other than eight positions of X, 1, or 0. See <i>SWITCH</i> in Part I.
11	NO SOURCE NAME GIVEN	Response to SOURCE was blank. See <i>SOURCE</i> in Part I.

Number	Message	Meaning
12	INVALID SOURCE NAME SPECIFIED	Response to SOURCE was invalid. See <i>SOURCE</i> in Part I.
13	INVALID DATE SPECIFIED	Response to DATE in file keywords was invalid. See <i>DATE (File Date)</i> in Part I.
14	TOO MANY RESPONSES TO A MULTI-VOLUME FILE KEYWORD	Only 10 volumes are allowed in each Multi-Volume File. See <i>Multi-Volume Files</i> in Appendix A.
15	NO FILE NAME GIVEN	Procedure contains file keywords but not FILE NAME response.
16	NO PACK GIVEN	Procedure contains file keywords but not PACK response.
17	INVALID FILE NAME SPECIFIED	Response to FILE NAME invalid. See <i>FILE NAME</i> in Part I.
18	INVALID LABEL SPECIFIED	Response to LABEL is invalid. See <i>LABEL</i> in Part I.
19	INVALID PACK SPECIFIED	Response to PACK is invalid. See <i>PACK</i> in Part I.
20	INVALID RETAIN DESIGNATION SPECIFIED	Response to RETAIN other than P, T, S or A. See <i>RETAIN</i> in Part I.
21	INVALID TRACKS SPECIFIED	See <i>RECORDS (and TRACKS)</i> in Part I.
22	MAXIMUM FILE STATEMENTS ENTERED	More than 15 file statements entered.

Number	Message	Meaning
23	BOTH TRACKS AND RECORDS SPECIFIED	Responses to both TRACKS and RECORDS have been given. See <i>RECORDS (and TRACKS)</i> in Part I.
24	INVALID RECORDS SPECIFIED	See <i>RECORDS (and TRACKS)</i> in Part I.
25	INVALID LOCATION SPECIFIED	Response to LOCATION must be 8 through 405.
26	DEVICE NOT SUPPORTED	CRT or READER referenced and not on system.
27	INVALID DEVICE	Response to DEVICE or READER invalid. See <i>Modify-Entering FORMS</i> in Part I.
28	INVALID NUMBER OF LINES	Response to LINES not between 12 and 112.
29	INVALID REQUEST	Response to MODIFY is invalid.
30	INVALID STATEMENT NUMBER	Invalid statement number entered as response to modify.
31	TOO MANY UTILITY CONTROL STATEMENTS IN PROCEDURE-JOB CANCELLED	
32	RUN OUT OF SPACE IN SCHEDULER WORK AREA	
33	RESPONSE REQUIRED-DELAYED RESPONSE IN CALLED PROCEDURE	
34	TOO MANY MULTI-VOLUME FILE UNITS SPECIFIED	Number of units specified exceeds number of packs specified.

Number	Message	Meaning
35	DELAYED RESPONSE (?) NOT ALLOWED	
36	JOB CANCELLED	You entered /* or job was cancelled because of errors.
37	MULTI-VOLUME NOT VALID THIS STATEMENT	Multi-response to this keyword is not allowed.
38	ENTER MINUS (-) NOT ALLOWED	ENTER- is allowed only during a BUILD cycle for some keywords.
39	ERRORS IN PROCEDURE – JOB CANCELLED	
40	ERRORS IN OCL STATEMENT	
41	ERRORS IN RESPONSE	
42	DUPLICATE PROCEDURE NAME IN LIBRARY	Response to BUILD NAME is already in source library of unit specified. See <i>BUILD NAME</i> in Part I.
43	DUPLICATE PROCEDURE DELETED	New procedure being entered will overlay old procedure with same name.
44	INVALID KEYWORD	Keyword found in procedure is invalid or response to READY is invalid.
45	TOO MANY UTILITY CONTROL STATEMENTS ENTERED	Only 40 utility control statements may be entered.
46	PERMANENT DISK ERROR	
47	RUN OUT OF SPACE IN PROCEDURE LIBRARY – JOB CANCELLED	

Number	Message	Meaning
48	INVALID SYSTEM DATE SPECIFIED	See <i>DATE (System Date)</i> in Part I.
49	DUPLICATE KEYWORD	A procedure contains a duplicate keyword.
50	RESPONSE REQUIRED	You must respond to this keyword. <i>PROG START</i> is not allowed.
51	TOO MANY PACKS, HIKEYS, OR BOTH SPECIFIED	Number of HIKEYS plus number of packs exceeds 52. Job cancelled.
52	DUPLICATE MULTI-VOLUME FILE UNIT SPECIFIED	See <i>Multi-Volume Files</i> in Appendix A.
53	INVALID RESPONSE DURING INQUIRY	Cannot change log device or change to card OCL.
54	INVALID HIKEY SPECIFIED	HIKEY entered is longer than <i>KEY LENGTH</i> specified or quotes not entered when copying single volume of multi-volume file.
55	INVALID HIKEY LENGTH SPECIFIED	Response to <i>KEY LENGTH</i> is greater than 29 or is 00.
56	HIKEYS OUT OF SEQUENCE	HIKEYS must be in ascending sequence.



IBM System/3 Model 6 users who have co-resident systems (both disk system management and System/3 BASIC) can transfer control from disk system management to System/3 BASIC by responding to READY with ENTER BASIC.

# APPENDIX J: IBM SYSTEM/3 STANDARD CHARACTER SET

Character	Hexadecimal Equivalent				
Blank	40		7E	W	E6
¢	4A	≠	7F	X	E7
.	4B	A	C1	Y	E8
<	4C	B	C2	Z	E9
(	4D	C	C3	0	F0
+	4E	D	C4	1	F1
	4F	E	C5	2	F2
&	50	F	C6	3	F3
↑	5A	G	C7	4	F4
\$	5B	H	C8	5	F5
•	5C	I	C9	6	F6
)	5D	}	D0	7	F7
:	5E	J	D1	8	F8
⌋	5F	K	D2	9	F9
- (minus)	60	L	D3		
/	61	M	D4		
.	6B	N	D5		
%	6C	O	D6		
- (underscore)	6D	P	D7		
>	6E	Q	D8		
?	6F	R	D9		
:	7A	S	E2		
#	7B	T	E3		
@	7C	U	E4		
' (Apostrophe)	7D	V	E5		

Capsule definitions of some common computer terms used in this manual.

<b>CPU</b>	(Central Processing Unit) Nucleus of the Model 6 hardware.
<b>end-of-job-halt</b>	system halt at the end of every job to give the operator time for any necessary housekeeping chores before beginning the next job.
<b>IPL</b>	(Initial Program Load) The process by which the operator loads into core storage the program that controls the operation of the system.
<b>KDE</b>	Keyboard Data Entry Utility Program
<b>KSE</b>	Keyboard Source Entry Utility Program
<b>object library</b>	contains compiled programs and system programs.
<b>object library directory</b>	contains name and address (track and sector) of each object program in the library.
<b>OCL</b>	(Operation Control Language) An OCL statement consists of a keyword and a response.
<b>overlay</b>	to erase data on disk by writing new data over it.
<b>procedure</b>	sequence of OCL statements in a source library.
<b>sector</b>	section of a disk track. Each track is divided into 24 sectors.
<b>source library</b>	contains procedures, RPG source programs, and KSE input.
<b>source library directory</b>	contains name and address (track and sector) of each source program in the library.
<b>source statements</b>	program instructions that have not been compiled (translated) into machine language.
<b>sysgen</b>	(system generation) Process required to get a system ready to run after installation.
<b>system printer</b>	displays OCL statements, utility control statements, job comments, and error codes. (The system printer can also display the normal output of the job being run.)
<b>track</b>	Each disk is divided into concentric circles called tracks.
<b>VTOC</b>	(Volume Table of Contents) That part of a disk which contains detailed information about every file on the disk.



- \$ALT (Alternate Track Assignment)
  - (see also alternate track assignment program)
  - as response to LOAD NAME in OCL cycle 53
- \$BUILD (Alternate Track Rebuild)
  - (see also alternate track rebuild program)
  - as response to LOAD NAME in OCL cycle 53
- \$COPY (Disk Copy/Dump)
  - (see also disk copy/dump program)
  - as response to LOAD NAME in OCL cycle 53
  - in OCL sample job #4 76
- \$DELET (File Delete)
  - (see also file delete program)
  - as response to LOAD NAME in OCL cycle 53
- \$DIU (Data Interchange Utility)
  - as response to LOAD NAME in OCL cycle 53
- \$DSORT (Disk Sort)
  - as response to LOAD NAME in OCL cycle 53
- \$INIT (Disk Initialization)
  - (see also disk initialization program)
  - as response to LOAD NAME in OCL cycle 53
  - in OCL sample job #1 70
- \$KDE (Keyboard Data Entry)
  - as response to LOAD NAME in OCL cycle 53
- \$KSE (Keyboard Source Entry)
  - as response to LOAD NAME in OCL cycle 53
- \$LABEL (File and Volume Label Display)
  - (see also file and volume label display program)
  - as response to LOAD NAME in OCL cycle 53
- \$MAINT (Library Maintenance)
  - (see also library maintenance program)
  - as response to LOAD NAME in OCL cycle 53
- \$RPG (RPG Compiler)
  - as response to LOAD NAME in OCL cycle 53
- \* (see comments)
- /& (card OCL) 223
- /\*
  - card OCL 223
  - conversational OCL 11
- // blank 88
- // ALLOCATE 137
  - (see also allocate, library maintenance)
- // ALT 102
  - (see also alternate track assignment program)
- // CEND 158
  - (see also copy, library maintenance)
- // COPY 158-162
  - (see also copy, library maintenance)
- // COPYPACK 132
  - (see also disk copy/dump program)
- // COPYFILE 132
  - (see also disk copy/dump program)
- // DELETE 177
  - (see also delete, library maintenance)
- // DISPLAY 115
  - (see also file and volume display program)
- // END
  - (see END control statement)
- // REBUILD 109
  - (see also alternate track rebuild program)
- // RENAME 179
  - (see also rename, library maintenance)
- // REMOVE 122
  - (see also file delete program)
- // SCRATCH 122
  - (see also file delete program)
- // SELECT KEY 132
  - (see also disk copy/dump program)
- // UIN 92
  - (see also disk initialization program)
  - in OCL sample job #1 70
- // VOL 92
  - (see also disk initialization program)
  - in OCL sample job #1 70
- ? (see delayed response)
- allocate, library maintenance
  - control statement summary 149
  - examples 182
  - parameter descriptions 151
  - parameter summary 150
  - uses 149
- ALT control statement 102
  - (see also alternate track assignment program)
- alternate track assignment
  - conditional assignment 104
  - unconditional assignment 105
  - cancel prior assignment 105
- alternate track assignment program 101
  - control statement summary 102
  - example 107
  - OCL considerations 106
  - parameter descriptions 104
  - parameter summary 103
  - program name 106
  - program uses 101
- alternate tracks
  - alternate track assignment 104
  - disk initialization 96
  - incorrect data on 105
- alternate track rebuild program 109
  - control statement summary 109
  - example 113
  - OCL considerations 112
  - parameter descriptions 111
  - parameter summary 110
  - program name 112
  - program uses 109
  - substitute data description 111
  - substitute data summary 110
- apostrophes in control statements 88, 193, 195
- asterisk
  - (see comments)
- ASSIGN parameter 105
- blanks in control statements 88
- BUILD NAME
  - in BUILD Keyword-Response Summary 27
  - its position in the BUILD cycle 26
  - keyword description 45

**BUILD cycle**  
 when to use 9

**BUILD NAME**  
 keyword description 205

**BUILD cycle**  
 when to use 204

**CALL NAME**  
 in the CALL Keyword-Response Summary 42  
 its position in the CALL cycle 41  
 keyword description 45

**CALL cycle**  
 when to use 9

**CANCEL**  
 as response to MODIFY in BUILD cycle 39  
 as response to MODIFY in CALL cycle 43  
 as response to MODIFY in LOAD cycle 24  
 effect of entering during BUILD cycle 59  
 effect of entering during CALL cycle 59  
 effect of entering during LOAD cycle 59  
 entering the keyword during MODIFY 59  
 cancelling alternate-track assignments 105  
 cancelling job  
   see MODIFY considerations in BUILD Keyword-Response Summary 39  
   see MODIFY considerations in CALL Keyword-Response Summary 43  
   see MODIFY considerations in LOAD Keyword-Response Summary 24

card OCL input 220-224

**CEND control statement**  
 reader-to-disk copy 158  
 disk-to-card copy 158

central processing unit (CPU)  
 definition 233

chained procedures 204

changing a previous OCL statement  
 during the MODIFY phase 56

changing file designation 52

changing object library size  
 control statement 150  
 disk considerations 154

changing printed output for system programs  
   see FORMS under MODIFY considerations in BUILD cycle 39  
   see FORMS under MODIFY considerations in CALL cycle 43  
   see FORMS under MODIFY considerations in LOAD cycle 24

changing size of source library  
 control statement 150  
 disk considerations 152

changing status of system printer  
   see LOG under MODIFY considerations in BUILD Keyword-Response Summary 39  
   see LOG under MODIFY considerations in LOAD Keyword-Response Summary 24  
   see MODIFY considerations in CALL Keyword-Response Summary 43

character set, standard 232

clear initialization 94

coding rules, control statements  
   use of apostrophes 88  
   use of blanks 88  
   use of commas 88  
   use of hyphens 88  
   statement length 88

commas in control statements  
   disk utilities 88

**OCL**  
   deleting statement 57  
   in HIKEY 199

**comments**  
   entering comments during the MODIFY phase 58

**COMPILE OBJECT**  
   in BUILD Keyword-Response Summary 29  
   in LOAD Keyword-Response Summary 17  
   its position in the BUILD cycle 26  
   its position in the LOAD cycle 15  
   keyword description 46

compiled RPG program  
   location of determined by OBJECT statement 46

compiling large RPG source programs 203

compiling RPG source programs  
   recommended method of 72

conditional assignment of alternate tracks 104

**control statement summaries**  
   alternate track assignment 102  
   alternate track rebuild 109  
   disk copy/dump 132  
   disk initialization 92  
   file and volume label display 115  
   file delete 121  
   library maintenance  
     allocate 150  
     copy 156  
     delete 177  
     rename 179

**control statements**  
   alternate track assignment  
     ALT statement 102  
   alternate track rebuild  
     REBUILD statement 109  
   coding rules 88  
   definition of disk/copy dump 88  
     COPYFILE statement 132  
     COPYPACK statement 132  
     SELECT statement 132  
   disk initialization  
     UIN statement 92  
     VOL statement 92  
   file and volume label display  
     DISPLAY statement 115  
   file delete  
     REMOVE statement 122  
     SCRATCH statement 122  
   library maintenance  
     ALLOCATE statement 137  
     COPY statement 158-162  
     DELETE statement 177  
     RENAME statement 179

**conversational OCL**  
   definition and how it works 7

**copy, library maintenance**  
   control statement summaries 158-162  
   examples 185  
   parameter descriptions 166  
   parameter summary 163  
   uses 156

**COPYFILE control statement 132**  
 copying disk from one removable disk to another on drive 1 136  
 copying entire disk 136  
 copying files 136

- copying library entries
  - reader-to-disk 166
  - disk-to-disk 168
- COPYPACK statement 132
- correcting OCL statements 56
- CPU (Central Processing Unit)
  - definition 233
  - requirements for conversational ii
- creating object library
  - control statement 149
  - disk considerations 153
- creating source library
  - control statement 149
  - disk considerations 151
- customer program name
  - as response to keyword LOAD NAME in OCL cycle 53
- DATA parameter 126
- DATA96
  - as response to keyword READER 22
- Data Interchange Utility (\$DIU)
  - as response to LOAD NAME in OCL cycle 53
- data recorder
  - used to code OCL statements on cards 220
- DATE (file date)
  - in BUILD Keyword-Response Summary 39
  - in LOAD Keyword-Response Summary 23
  - keyword description of 52
  - position in BUILD sequence 26
  - position in LOAD sequence 15
  - restrictions during file creation runs 52
- DATE parameter
  - file delete program 126
- DATE statement, format of
  - definition 47
  - general restrictions 47
- DATE (system date)
  - in BUILD Keyword-Response Summary 31
  - in LOAD Keyword-Response Summary 18
  - keyword description 47
  - position in BUILD sequence 26
  - position in LOAD sequence 15
- defective tracks
  - address on disk 104
  - definition (see surface analysis)
  - retesting of 95
- delayed response
  - definition of, restrictions, effect on system 9
- delayed responses in procedure
  - see footnote 1B of CALL Keyword-Response Summary
- delete, library maintenance
  - control statement summary 177
  - examples 188
  - parameter summary 178
  - uses 176
- DELETE parameter 177
- deleting a previous OCL statement
  - during the MODIFY phase 57
- deleting files 126
- deleting library entries 176
- deleting object library
  - control statement 150
  - disk considerations 154
- deleting procedures
  - general discussion 45
- deleting records from a file 137
- deleting source library
  - control statement 150
  - disk considerations 152
- designation of library entry 174
- direct files
  - deleting records from 137
  - OCL consideration for multi-volume files 196
  - printing part of 138
  - records-tracks conversion for 212
- disk copy/dump program
  - control statement summary 132
  - examples 143
  - considerations, OCL 140
  - copying entire disk 136
  - copying or printing files 136-137
  - parameter descriptions 136
  - parameter summary 134
  - program name 140
  - program uses 131
- disk drive
  - capacity 94
  - requirements for conversational OCL ii
- disk files 213
- disk initialization program 91
  - control statement summary 92
  - example 98
  - OCL considerations 97
  - parameter descriptions 94
  - parameter summary 93
  - program name 97
  - program uses 91
- disk name
  - characters allowed in 96
  - length of 95
  - response to PACK in OCL cycle 50
  - uses
    - alternate track assignment 104
    - alternate track rebuild 111
    - disk initialization 96
    - file delete 126
- disk organization 213
- disk-to-card copy
  - considerations 172
  - control statements 161
- disk-to-disk copy
  - considerations 168
  - control statements 158
- disk to printer and card copy
  - considerations 172
  - control statements 162
- disk-to-printer copy
  - considerations 167
  - control statements 160
- DISP (displacement) parameter 111
- DISPLAY control statement 115
- duplicate procedure names
  - general discussion 45
  - operator's options following 45
- END control statement 89
- end-of-job halt
  - definition 233
- ENTER - Key
  - purpose of, when to use 13
  - use in bypassing non-required file keywords 13
  - uses of 13

- ENTER + Key
  - its function and its relationship to the PROG START 13
  - key
  - purpose of, when to use 13
  - uses of 13
- entering comments
  - during the MODIFY phase 58
- error code
  - (see error messages)
- error code options 2
- error messages 225
- errors in OCL statements
  - how to correct using MODIFY statement 56
- examples
  - alternate track assignment
    - conditional assignment 107
  - alternate track rebuild
    - correcting characters on alternate track 113
  - disk copy/dump
    - copying entire disk 143
    - copying a file 144
    - printing part of a file 148
  - disk initialization
    - primary initialization 98
  - file and volume label display
    - printing VTOC information for two files 120
  - file delete
    - deleting one of several files having same name 128
  - library maintenance
    - changing source library size 183
    - copying minimum system 185
    - creating libraries 182
    - deleting object library 184
    - deleting permanent entries of one type 190
    - deleting temporary entry 188
    - deleting temporary entries with names beginning with certain characters 189
    - printing library directories 186
    - renaming source statements 191
    - replacing library entry 187
- OCL
  - chained procedures 209
  - compile RPG II source 72
  - copy disk 76
  - include utility control statements in procedure 202
  - initialize a disk 70
  - multi-file CALL 81
  - multi-file BUILD 78
  - multi-volume indexed file creation 199
  - multi-volume indexed file update 198
  - process customer program 74
- external indicators
  - at IPL 64
  - considerations when responding to SWITCH in BUILD cycle 66
  - considerations when responding to SWITCH in LOAD cycle 65
  - current setting displayed in SWITCH statement 64
  - using the SWITCH statement to change 64
- file and volume label display program
  - control statement summary 115
  - example 120
  - OCL considerations 119
  - parameter descriptions 117
  - parameter summary 116
  - program name 119
  - program uses 115
- file date
  - keyword description 52
  - restriction during file creation run 52
- file dates 126
- file delete program
  - control statement summary 122
  - examples 128-129
  - OCL considerations 127
  - parameter descriptions 126
  - parameter summary 124
  - program name 127
  - program uses 121
- file designation
  - how to change 52
  - response to RETAIN in OCL cycle 51
- file keywords
  - system-operator interaction during prompting of 49
- FILE NAME
  - for \$DSORT, \$COPY, \$MICR, \$RPG, and \$KDE 48
  - for RPG Programs
    - in BUILD Keyword-Response Summary 32
    - in LOAD Keyword-Response Summary 19
  - its position in the BUILD sequence 26
  - its position in the LOAD sequence 15
  - keyword description 48
- file names
  - file delete 126
  - disk copy/dump 136
- files, direct
  - records-tracks conversion for 212
- files, indexed
  - records-tracks conversion for 212
- files, multi-volume
  - OCL considerations for 194, 224
- files, sequential
  - records-tracks conversion for 212
- FORMS
  - entering the keyword during the MODIFY phase 60
- FROM parameter
  - disk copy-dump
    - copying entire disk 136
    - copying or printing files 136
  - library maintenance 166
- glossary 233
- HALT
  - in card OCL 221, 224
  - in conversational OCL 52
- halt, end-of-job
  - definition 233
- HIKEY (see multi-volume files)
- how to use this manual 1
- hyphens in control statements 88
- IBM System/3 standard character set 232
- IBM-Supplied RPG Compile Procedure (RPG)
  - as response to CALL NAME in CALL sequence 45
  - increasing size of 203
  - in sample job #2 72
- ID (identification) parameter 96
- INCLUDE
  - during a CALL cycle 63
  - entering during the MODIFY phase 62
  - including control statements in a procedure 202



- response to MODIFY in BUILD sequence 40
- restrictions following keyword 63
- sample job 202
- special considerations involving INCLUDE statements 63
- indexed files
  - multi-volume
    - file statements for 198-224
    - OCL considerations for 196
    - OCL sample jobs for 198-199
    - printing part of 149
    - record-tracks conversion for 212
    - reorganizing 137
- initial program load (IPL)
  - definition 223
  - establishing system date at 46
- incorrect data on alternate tracks 105
- initialization
  - clear initialization 94
  - general definition 91
  - primary initialization 94
  - secondary initialization 94
- KEY LENGTH (see multi-volume files)
- keyword 7
- keyword descriptions
  - for each keyword 45-67
  - what they are and how to use them 5
- keyword flowcharts
  - what they are and how to use them 5
- keyword prompting
  - how it's done 7
- keyword-response summary
  - for the BUILD sequence 27
  - for the CALL sequence 42
  - for the LOAD sequence 16
- keyword-response summaries
  - what they are and how to use them 5
- LABEL parameter
  - File and volume label display 116
  - File delete 126
  - OCL
    - in BUILD Keyword-Response Summary 34
    - in LOAD Keyword-Response Summary 21
    - its position in the BUILD sequence 26
    - keyword description 50
    - position in LOAD sequence 15
    - when response is required 50
- large RPG programs, compiling 203
- LENGTH parameter 111
- length on control statements 88
- library directories
  - definitions 147
  - directory printouts 173
  - object library directory size 153
  - source library directory size 151
- library entries
  - choosing designation 174
  - copying entries
    - considerations 166-175
    - control statements 158-162
  - deleting entries 177
  - naming entries 173
  - organization in libraries 147
  - renaming entries 179
  - types 147, 172
- library maintenance program
  - control statement summaries
    - allocate 150
    - copy 158-162
    - delete 177
    - rename 179
  - examples
    - allocate 183
    - copy 185
    - delete 188
    - rename 191
  - library description 147
  - OCL considerations 182
  - parameter descriptions
    - allocate 151
    - copy 166
  - parameter summaries
    - allocate
    - copy 163
    - delete 179
    - rename 180
  - program name 181
  - program uses
    - allocate 149
    - copy 156
    - delete 176
    - rename 179
- library, object
  - definition of 233
- LIBRARY parameter 172
- library, source
  - definition 233
- line counter specifications
  - (see FORMS)
- LOAD NAME
  - in BUILD Keyword-Response Summary 27
  - in LOAD Keyword-Response Summary 16
  - its position in BUILD sequence 26
  - its position in LOAD sequence 15
  - keyword description 53
- LOAD sequence
  - when to use 9
- LOCATION
  - considerations for multi-volume files 196
  - in the BUILD Keyword-Response Summary 37
  - in the LOAD Keyword-Response Summary 22
  - its position in BUILD sequence 25
  - its position in LOAD sequence 15
  - keyword description 51
- location of libraries on disk
  - source with respect to object 147
  - placement of source library 151
  - placement of object library 153
- LOG
  - 22" printer as log device 214
  - CRT as log device 216
  - entering during MODIFY 218
  - entering during READY 217
- machine requirements ii
- Model 6 disk organization 213
- Model 6 job cycle 7
- MODIFY
  - changing a previous OCL statement 56
  - deleting a previous OCL statement 57
  - entering CANCEL 59

**MODIFY (continued)**  
 entering comments 58  
 entering FORMS 60, 215  
 entering INCLUDE 63  
     restrictions on 63  
 entering LOG 218  
 in BUILD Keyword-Response Summary 39  
 in CALL Keyword-Response Summary 43  
 in LOAD Keyword-Response Summary 24  
 its position in the BUILD cycle 26  
 its position in the CALL cycle 41  
 its position in the LOAD cycle 15  
 keyword description of MODIFY options 54  
 statement numbers 13  
 multiple files 50  
 multi-volume files 193  
     coding for 198  
     OCL considerations for 196  
     sample jobs 198-199  
  
 name of source program  
     as response to COMPILE SOURCE in OCL BUILD cycle 46  
     as response to COMPILE SOURCE in OCL LOAD cycle 46  
 NAME parameter 173  
 naming library entries 173  
     characters to use 173  
     length of name 173  
     restricted names 173  
 NEWNAME parameter 174  
 NOHALT  
     in card OCL 221, 224  
     in conversational OCL 63  
  
 object library  
     changing size  
         control statement 149  
         disk considerations 154  
     creating  
         control statement 149  
         disk considerations 153  
     definition 147  
     deleting  
         control statement 149  
         disk considerations 154  
     reorganizing  
         control statement 149  
         disk considerations 154  
 object library directory  
     definitions 148, 172  
     printout 167  
     size 153  
 OBJECT parameter 153  
 object programs, definitions of 147, 172  
 OCL  
     definition 7, 233  
 OCL considerations  
     alternate track assignment 106  
     alternate track rebuild 112  
     disk copy/dump 140  
     disk initialization 97  
     file and volume label display 119  
     file delete 127  
     library maintenance 181  
     multi-volume files 196  
 OCL cycle 7-9  
 OCL guide  
     sample form 219  
  
 operation control language (OCL)  
     definition of 7, 233  
 operator's OCL guide  
     sample form 219  
 organization of library entries 147  
 OUTPUT parameter 136  
 OUTPTX parameter 136  
 overlay  
     definition 233  
 overriding system date 47  
  
 P (permanent) file designation  
     importance in deleting a procedure from a source library 45  
 P (permanent) files  
     restrictions 51  
     when to assign a P (permanent) designation to a file 51  
 PACK parameter  
     alternate track assignment 104  
     alternate track rebuild 111  
     disk initialization 96  
     file delete 126  
     OCL 50  
     considerations for multi-volume files 194  
     in BUILD Keyword-Response Summary 34  
     in LOAD Keyword-Response Summary 21  
     its position in BUILD sequence 26  
     its position in LOAD sequence 15  
     keyword description 50  
 parameter 88  
 parameter descriptions  
     alternate track assignment 104  
     alternate track rebuild 111  
     disk copy/dump 136  
     disk initialization 94  
     file and volume label display 116  
     file delete 126  
     library maintenance  
         allocate 151  
         copy 166  
     parameter summaries  
         alternate track assignment 103  
         alternate track rebuild 110  
         disk copy/dump 134  
         disk initialization 93  
         file and volume label display 116  
         file delete 124  
         library maintenance  
             allocate 150  
             copy 163  
             delete 178  
             rename 180  
 permanent (P) files  
     restrictions 51  
     when to assign a P (permanent) designation to a file 51  
 predefined filenames  
     for \$DSORT, \$COPY, \$MICR, \$RPG, and \$KDE programs 48  
 primary initialization 94  
 primary tractor  
     in entering LOG during the MODIFY phase 214  
     lines per page setting for 214  
     print positions of 214  
 printing entire VTOC 116  
 printing file information from VTOC 116  
 printing files 137  
 permanent library entries 174  
 printing library directories 166-173  
 printing library entries 166-173

printing part of an indexed file 138  
 printing part of direct file 138  
 printing part of sequential file 138  
 procedure  
   definition of 147, 172, 233  
   deleting 45  
 procedure name  
   as response to CALL NAME in CALL cycle 45  
   response to BUILD NAME in BUILD cycle 45  
   restrictions on 45  
**PROG START** key  
   uses of 13  
   (see also keyword-response summary)  
   when to use it 13  
 program names  
   alternate track assignment (\$ALT) 106  
   alternate track rebuild (\$BUILD) 112  
   disk copy/dump (\$COPY) 140  
   disk initialization (\$INIT) 97  
   file and volume label display (\$LABEL) 119  
   file delete (\$DELETE) 127  
   library maintenance (\$MAINT) 181  
 program operation 87  
   all programs except library maintenance 87  
   library maintenance 88  
 prompting  
   how it's done 7  
 punching library entries 172  
  
 question mark key  
   purpose 9  
  
 reader-to-disk copy  
   considerations 168  
   control statements 158  
**READY**  
   in BUILD Keyword-Response Summary 27  
   in CALL Keyword-Response Summary 42  
   in LOAD Keyword-Response Summary 16  
   its position in the BUILD sequence 26  
   its position in the CALL sequence 41  
   its position in the LOAD sequence 15  
   its position in the Model 6 job cycle 7  
   keyword description 64  
**REBUILD** 110  
**RECORDS**  
   considerations for multi-volume files 196  
   in BUILD Keyword-Response Summary 35  
   in LOAD Keyword-Response Summary 21  
   its position in the BUILD sequence 26  
   its position in the LOAD sequence 15  
   keyword description 50  
 records-track conversion 212  
 relative record number 138  
**REMOVE** statement 122  
 rename, library maintenance  
   control statement summary 179  
   example 191  
   parameter summary 180  
   use 179  
 renaming library entries 179  
**REORG** (reorganize) parameter 137  
 reorganizing indexed files 137  
 reorganizing object library  
   control statement 149  
   disk considerations 154  
   reorganizing source library  
     control statement 149  
     disk considerations 153  
 replacing library entries  
   reader-to-disk copy 166  
   disk-to-disk copy 168  
   RETAIN parameter 174  
   NEWNAME parameter 175  
**RETAIN** parameter  
   library maintenance program 174  
**OCL**  
   in BUILD Keyword-Response Summary 38  
   in LOAD Keyword-Response Summary 23  
   its position in BUILD sequence 26  
   its position in LOAD sequence 15  
   key description 51  
**RPG Compiler (\$RPG)**  
   as response to LOAD NAME in OCL cycle 53  
**RPG File Description Specifications**  
   source of RPG Filename in OCL cycle 48  
**RPG filename**  
   response to FILE NAME in OCL cycle 48  
**RPG programs**  
   compiling 72  
   compiling large RPG programs 203  
   recommended method of compiling 72  
**RPG source programs**  
   compiling 72  
   compiling large RPG source programs 203  
   recommended method of compiling 72  
**RUN**  
   keyword description 64  
   response to MODIFY in BUILD sequence 39  
   response to MODIFY in CALL sequence 43  
   response to MODIFY in LOAD sequence 24  
   routines, definitions of 147, 172  
  
**S (scratch) files**  
   restrictions 52  
   when to apply an S (scratch) designation to a file 51  
   (see examples)  
 schedular work area 153  
**SCRATCH** control statement 122  
 scratch (S) files  
   restrictions 52  
   when to apply an S (scratch) designation to a file 51  
 secondary initialization 94  
 secondary tractor (of 22" printer)  
   entering LOG for 214  
   lines per page setting for 214  
 sector  
   definition 233  
**SELECT** control statement 132  
**SELECT KEY** parameters 138  
**SELECT PKY** parameters 138  
**SELECT RECORD** parameters 138  
 sequential files  
   deleting records from 137  
   printing part of 138  
   records-tracks conversion for 212  
 sequential multi-volume files  
   OCL considerations for 196  
 setting external indicators 64  
 single quotation mark key  
   (see multi-volume file)  
**SORT** source statements in a procedure  
   (see footnote 1A in CALL Keyword-Response Summary)

## SOURCE

- in BUILD Keyword-Response Summary 29, 46
- in the LOAD Keyword-Response Summary 18, 46
- its position in the BUILD sequence 26
- its position in the LOAD sequence 15
- keyword description 46
- source library
  - changing size
    - control statement 149
    - disk considerations 152
  - contents 213
  - creating
    - control statement 149
    - disk considerations 151
  - definition 147, 233
  - deleting
    - control statement 149
    - disk considerations 152
  - its relationship to the BUILD and CALL sequences 9
  - putting procedures in 45
  - reorganizing
    - control statement 149
    - disk consideration 153
- source library directory
  - definitions 148, 172, 233
  - printout 167, 186
  - putting procedure names in 45
  - size 151
- SOURCE parameter 151
- source statements
  - as input to the RPG Compiler 203
  - definition 147, 172, 233
- source unit 47
- special characters
  - their uses and location 85
- standard character set 232
- statement numbers 13
  - in modify 54
- status of system printer
  - consideration when responding to MODIFY with a LOG statement 214
- substitute data 111
- surface analysis
  - alternate track assignment 104
  - disk initialization 95

## SWITCH

- in BUILD cycle 66
- in CALL cycle 67
- in LOAD cycle 65
- its position in BUILD sequence 26
- its position in LOAD sequence 15
- keyword description 64

### SWITCH Statement

- during a BUILD cycle 66
- during a CALL cycle 67
- during a LOAD cycle 65

## sysgen

- definition 233

## system date

- keyword description 47
- overriding 47
- responding to in the BUILD sequence 30
- responding to in the LOAD sequence 18

## system director

- definition 148
- printout 166

- system input device
  - general use 87
  - use in library maintenance 166
- system-operator interaction during keyword prompting 11
- SYSTEM parameter 155
- system printer
  - definition 233
  - (see also FORMS and LOG)
- system program name
  - as response to keyword LOAD NAME in OCL cycle 53
- system programs — changing printed output for (see FORMS under MODIFY)
- system programs, including in object library 155

## T (temporary) Files

- restrictions 52
- when to assign a T (temporary) designation to a file 51

## temporary (T) files

- restrictions 52
- when to assign a T (temporary) designation to a file 51

## temporary library entries 174

## testing condition of disk tracks (see surface analysis)

## TO parameter

- disk copy/dump
  - copying entire disk 136
  - copying or printing files 136-137
- library maintenance
  - allocate 151
  - copy 166

## TRACK parameter 111

## TRACKS

- considerations for multi-volume files 196
- definition 233
- in BUILD Keyword-Response Summary 35
- in LOAD Keyword-Response Summary 21
- its position in the BUILD sequence 26
- its position in the LOAD sequence 15
- keyword description 50-51
- tracks-records conversion 212

## TYPE parameter 94

## types of library directories 172

## types of library entries 172

## UIN control statement 92

## UNASSIGN parameter 105

## unconditional assignment of alternate tracks 105

## UNIT parameter

- alternate track assignment 104
- alternate track rebuild 111
- disk initialization 94
- file and volume label display 116
- file delete 126

## OCL

- BUILD unit 26
- FILE unit 50
- keyword description 34
- LOAD unit 15
- multi-volume files 196
- SOURCE unit 47

## utility control statements in procedure (see BUILD cycle)

## VERIFY parameter

- alternate track assignment 104
- disk initialization 95

## VOL control statement 92

**VTOC (volume table of contents)**

- contents 213
- definition 116, 233
- its relationship to LABEL 50
- printing entire VTOC 116
- printing file information only 116

**VTOC file name**

- as response to keyword LABEL in OCL cycle 50
- how to distinguish two files with the same VTOC file name and label 52

**work area**

- disk copy/dump 137
- library maintenance
  - allocate function 154-155
  - scheduler 153

**WORK parameter**

- disk copy/dump 137
- library maintenance 154

**1255 Magnetic Character Reader Utility (\$MICR)**

- in response to LOAD NAME in OCL cycle 53

**13 inch printer**

- requirements for conversational OCL ii

---





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

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



## READER'S COMMENT FORM

IBM System/3  
Model 6  
Operation Control Language and  
Disk Utility Programs  
Reference Manual

GC21-7516-2

### YOUR COMMENTS, PLEASE. . .

Your comments concerning this publication will help us produce better publications for your use. Each reply will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

*Note:* Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.

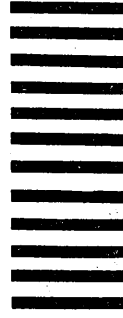
Cut Along Line

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  
Rochester, Minnesota 55901

Attention: Programming Publications, Dept. 425

Fold

Fold



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

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