IBM System/3 Model 6 System Programmer's Guide

IBM System/3
Model 6 System
Programmer's Guide

First Edition (July 1971)

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

Address comments concerning the content of this publication to IBM Corporation, Programming Publications, Department 425, Rochester, Minnesota 55901.

PREFACE

This manual assumes that you have had little data processing training or experience. You should, however, be familiar with the IBM System/3 Model 6 as it is presented in the *IBM System/3 Model 6 Introduction,* GA21-9122.

The purpose of this manual is to instruct you in:

- Why all jobs require OCL statements.
- The function of each OCL statement.
- The general purpose of the disk utility programs.
- The major functions and options for each disk utility program.

This should provide you with the background material needed to use the *IBM System/3 Model 6 Operation Control Language and Disk Utility Programs Reference Manual,* GC21-7516.

A series of review questions provided in Part V of this manual is for the OCL section of this manual. You should use these questions to check your understanding of the important concepts in Part II.

Referenced Publications

The following publications are referenced in this manual:

- *IBM System/3 Model 6 Operation Control Language and Disk Utility Programs Reference Manual,* GC21-7516
- *IBM System/3 Model 6 Halt Procedure Guide,* GC21-7541
- *IBM System/3 Disk Sort Reference Manual,* SC21-7522

This manual is divided into five parts:

Part I. System Summary. This part contains a description of the basic components of the Model 6.

Part II. Conversational OCL. This part contains a description of OCL (operation control language) for the Model 6 including program, compile, and file keywords, the four OCL cycles, error messages, and compiling RPG II programs. Part II is divided into 12 chapters. These chapters should be read sequentially.

Part III. Disk Utility Programs. This part contains descriptions of the disk utility programs: Disk Initialization, Alternate Track Assignment, Alternate Track Rebuild, File and Volume Label Display, File Delete, Disk Copy/Dump, and Library Maintenance. Part III is divided into eight chapters: an introductory chapter and a chapter for each utility program. Each utility chapter is divided into four sections: introduction, function, options, and control statements.

Part IV. Sample Jobs. This part provides six examples of jobs similar to what you might run. Each sample job is divided into three parts: introduction, statements, and explanation.

Part V. Review Questions. This part contains review questions for each chapter in Part II of this manual. Use it to check your understanding of the important concepts in each chapter. Part V is divided into two sections: questions and answers.

In addition, this manual contains a glossary and an appendix. The glossary furnishes definitions of both terms that are defined in text and those that are not. Refer to it when you find a term you are not familiar with or to refresh yourself on the exact meaning of a term. *Appendix A. Operator's OCL Guide* describes the document you may use to relay your information concerning keyword responses to the operator. The operator uses this document to key in your responses to the system prompts.

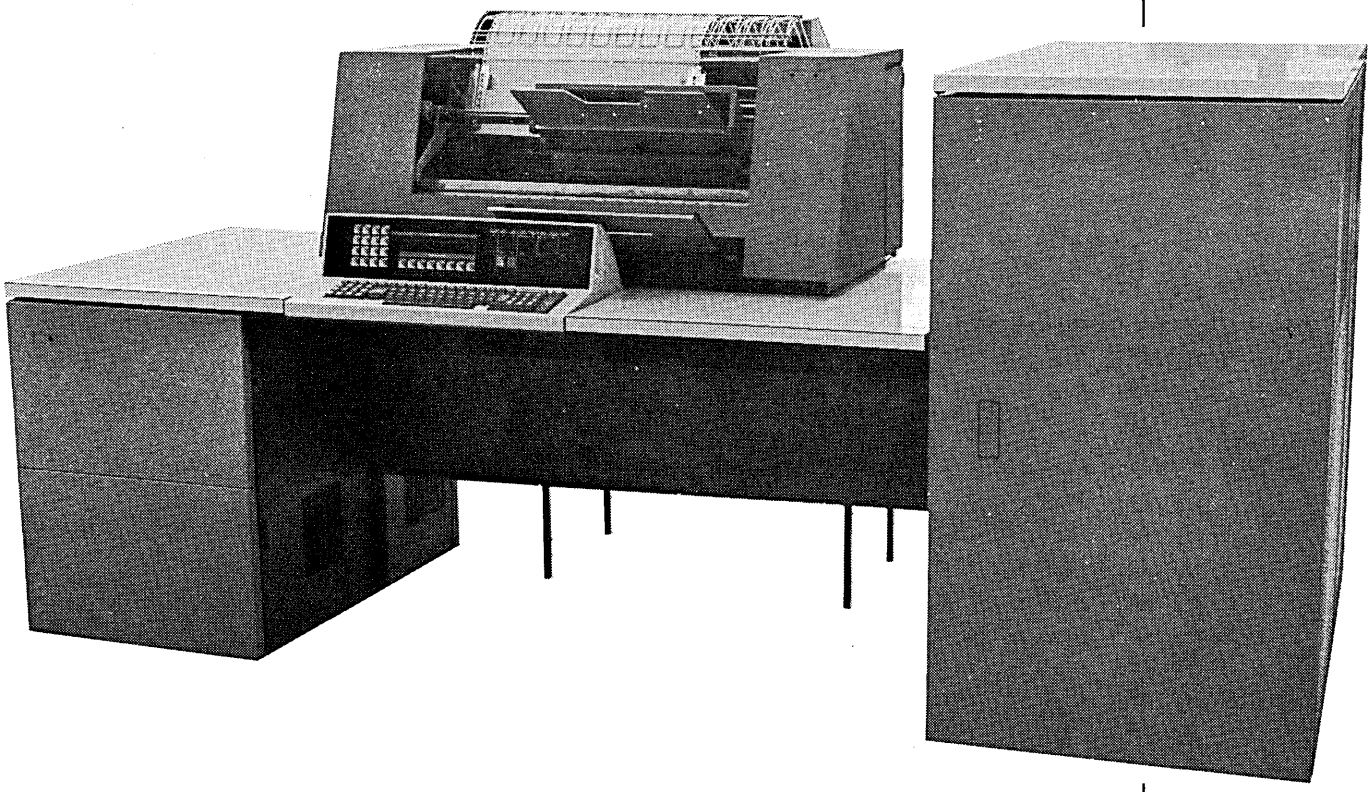
CONTENTS

PART I. SYSTEM SUMMARY

CHAPTER 1. BASIC COMPONENTS

The IBM System/3 Model 6 is designed to meet the requirements of business data processing. You might use it to perform any of the following business applications:

Order writing
Billing
Accounts receivable
Accounts payable
Inventory control
Payroll
Sales analysis



53978

The standard units of the Model 6 are the:

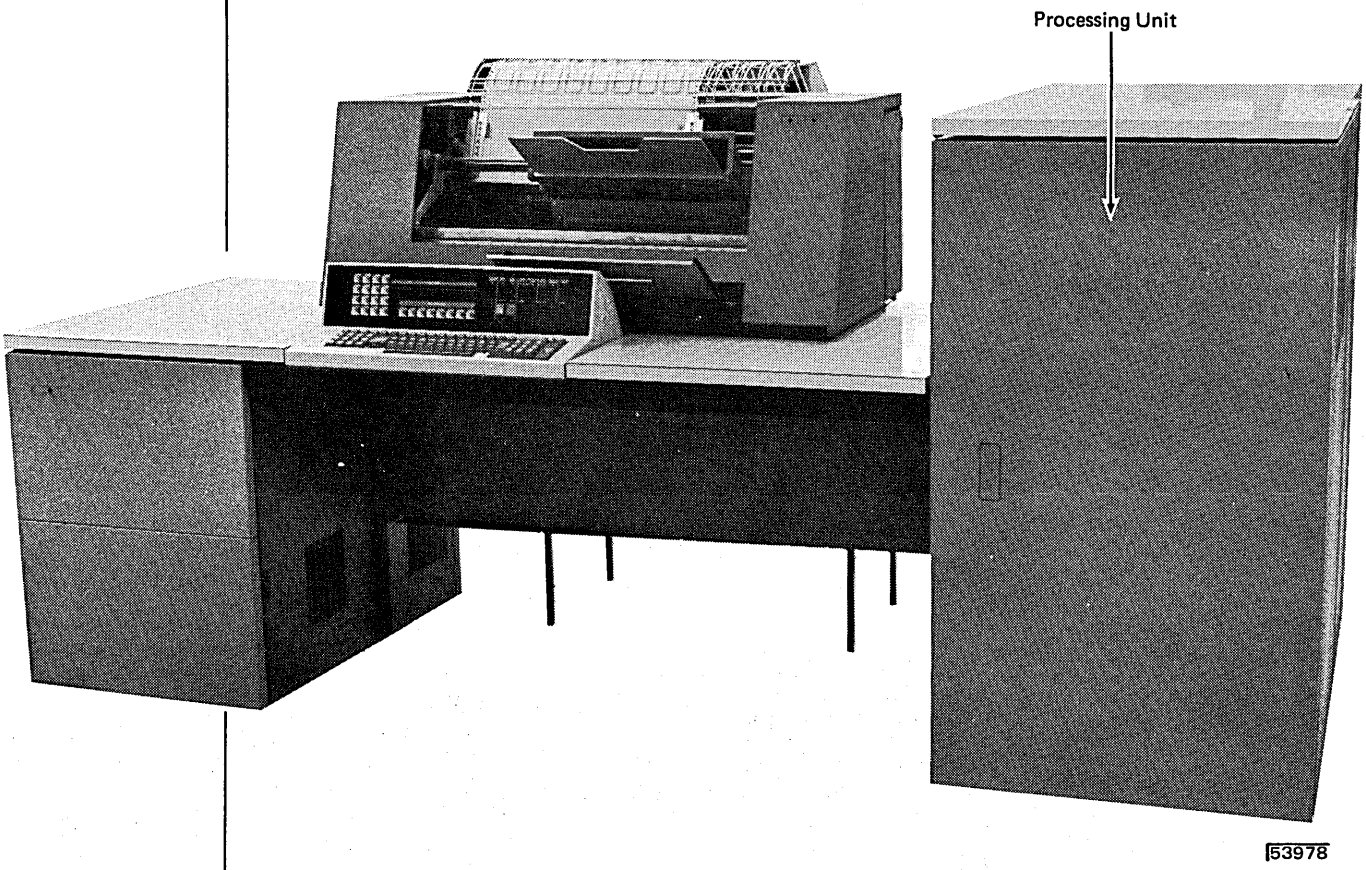
IBM 5406 Processing Unit with Operator Keyboard Console
IBM 5444 Disk Storage Drive
IBM 5213 Printer

Optional units of the Model 6 are the:

IBM 5496 Data Recorder
IBM 2222 Printer
IBM 2265 Display Station (Cathode Ray Tube)

IBM 5406 PROCESSING UNIT

The processing unit provides the control, arithmetic, and logical functions for the system, as well as storage for instructions and data.

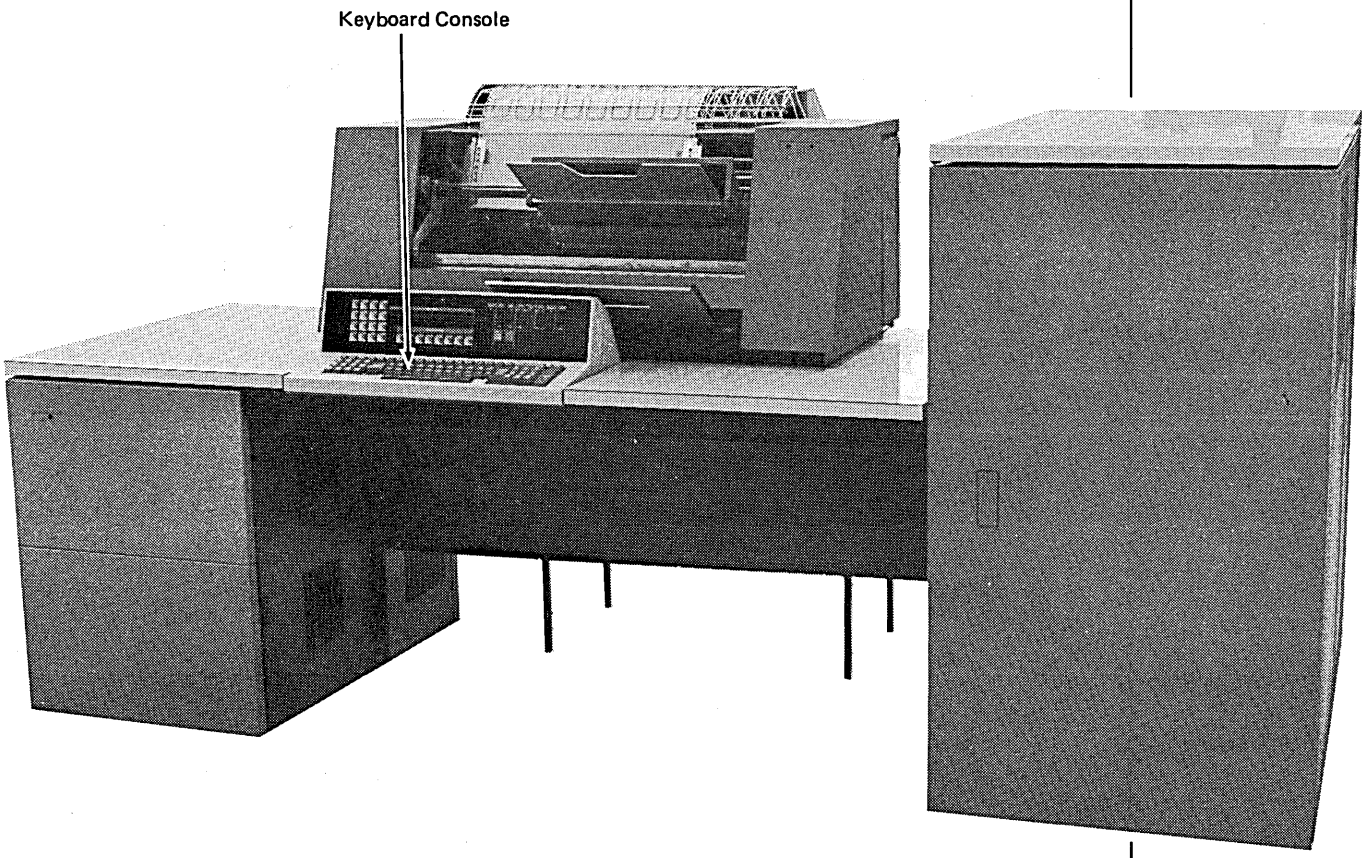


The basic unit of storage is the *byte.* Each byte will store one alphabetic character, one special character, or two numeric digits of information. Bytes may be handled separately or grouped together to form *fields.*

All processing of data is carried out in the processing unit under control of the program instructions. The processing unit controls all input/output (I/O) devices attached to the system. The I/O operations performed and devices used are specified by program instructions.

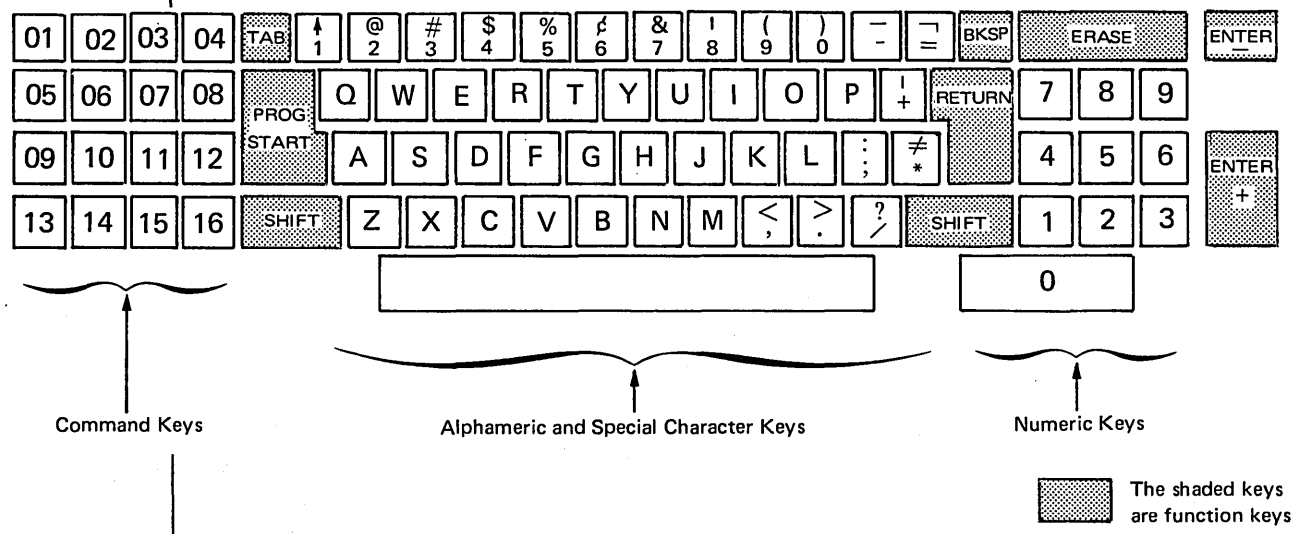
OPERATOR KEYBOARD CONSOLE

The operator keyboard console provides two-way communication between the operator and the system. When entering data into the system and controlling the operation of the system, the operator uses the operator keyboard, the system display panel, and the system control panel, all located on the operator console.



53978

Operator Keyboard



The shaded keys are function keys

The operator keyboard is the device the operator uses most often for entering information into the Model 6. The operator can enter:

- Operation control language (OCL) and utility control statements
- RPG II source programs
- Disk Sort specifications
- Input data to user or system programs

The keyboard uses *keys* to perform certain functions such as spacing and backspacing. There are four groups of keys:

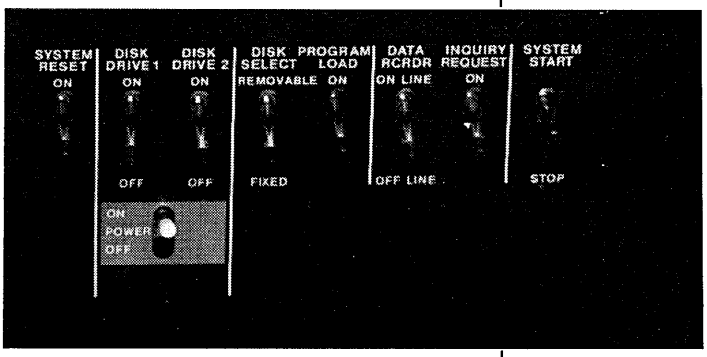
Command Keys. These keys allow the operator to control operations performed by RPG II programs and the Model 6 conversational utility programs.

Function Keys. These keys allow the operator to control certain printer operations and to perform required program functions, such as designating the end of a keying operation or erasing fields from storage. (The end-of-statement keys discussed in *Chapter 2. End-of-Statement Keys* in *Part II* of this manual are an example of function keys.)

Alphameric and Special Character Keys. These keys allow the operator to enter alphameric data or control information into the system.

Numeric Keys. These keys allow the operator to enter numeric data into the system. They are used when the data to be keyed in is primarily numeric.

System Control Panel



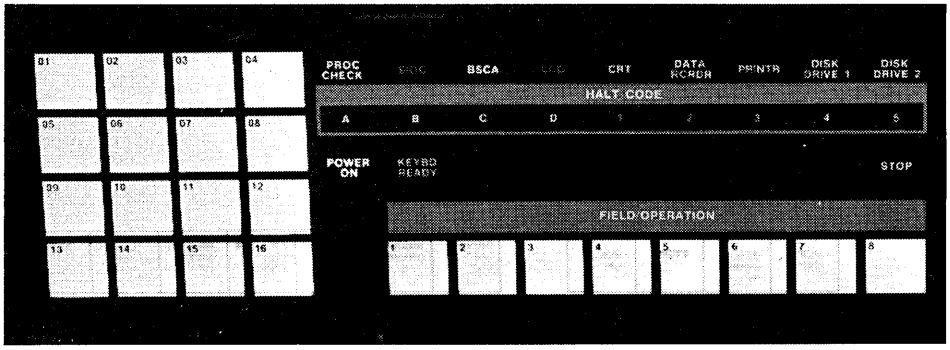
53979

The operator uses the switches on the system control panel, along with controls and indicators on the Model 6 devices, to control the operation of the system. Examples of the switches on the panel are:

SYSTEM START/STOP Switch. When moved to START, this switch allows the system to continue normal operation. This move is made only if STOP has been indicated. STOP causes the system to stop after it completes the operation currently in process.

POWER ON/OFF Switch. This switch controls power to the units of the system.

System Display Panel



53979

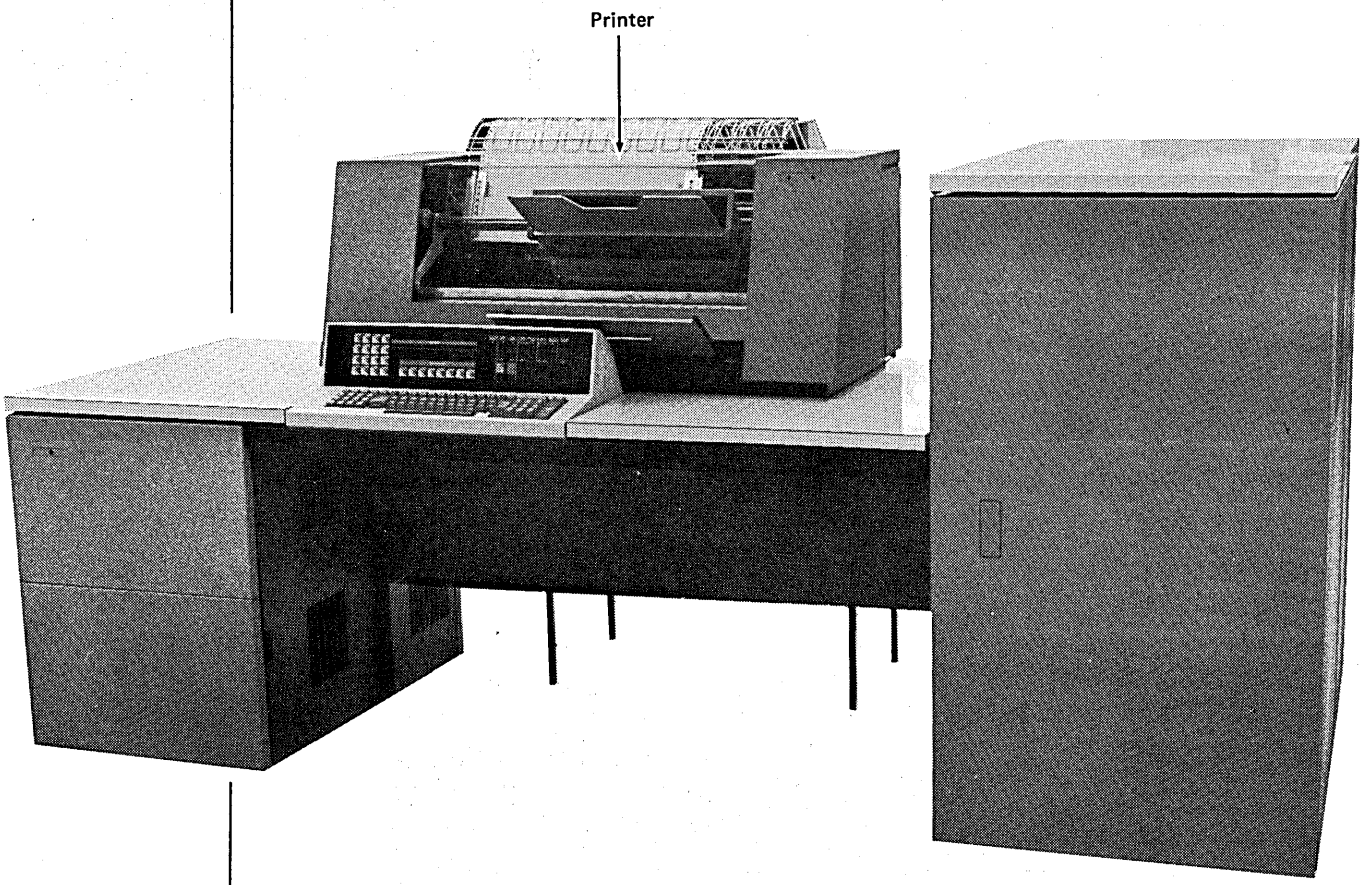
The indicator lights on this panel indicate the status of the system. Examples of the information contained on the panel are:

Halt Code Display. This unit displays characters when certain program halts occur. (The *IBM System/3 Model 6 Halt Procedure Guide,* GC21-7541, contains further information on halts.) The displayed characters are used to identify the halt.

Command Key Lights. When a command key is pressed, a light on the panel turns on. Each command key has a light to help the operator remember which command keys are on. This light won't go off until after the command key is turned off by the system.

IBM 5213 PRINTER

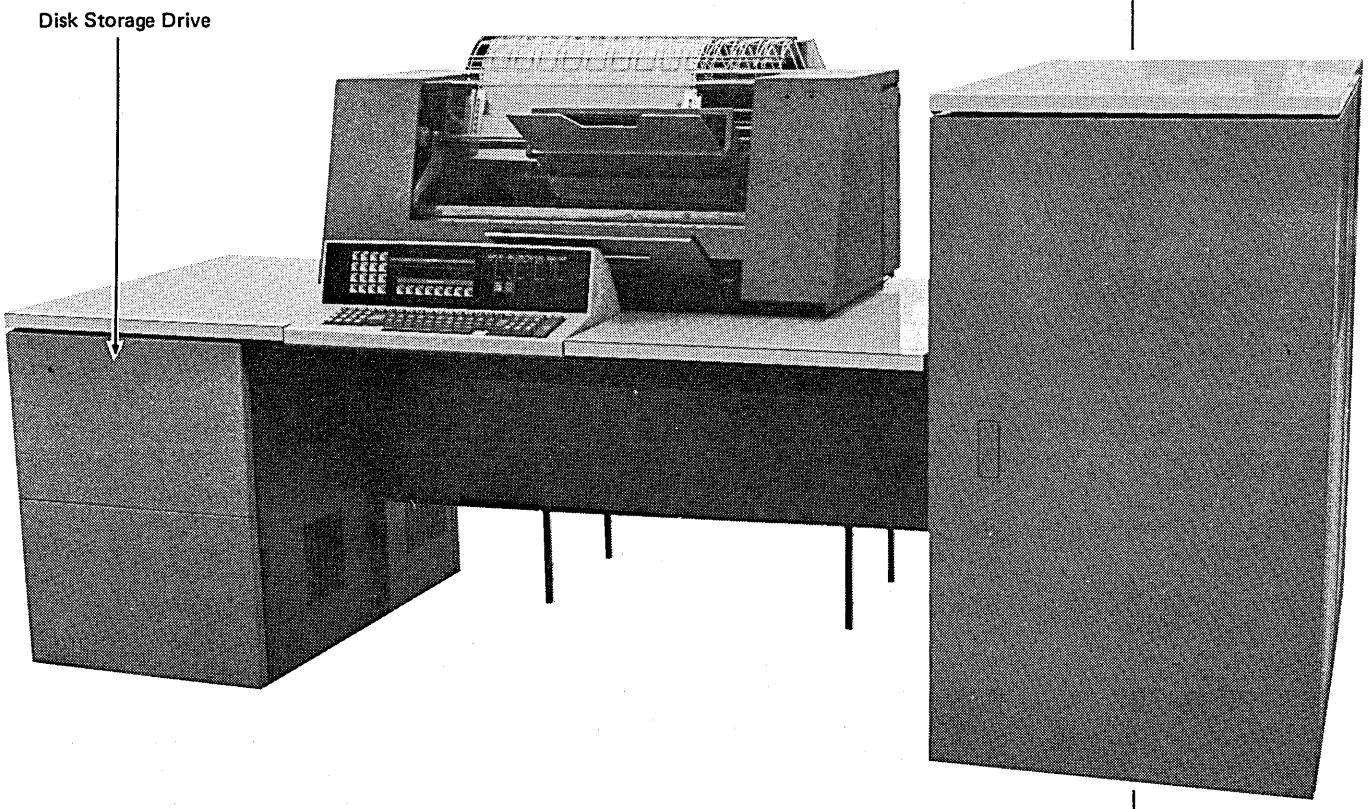
The 5213 printer is a serial printer which has all printer operations controlled by the program in storage. The operator's instructions to the system are printed as he keys them. The system's reply is also printed, providing easy reference for the operator. The printer also provides output of the results of a program in the form of printed reports controlled by the program instructions.



53978

IBM 5444 DISK STORAGE DRIVE

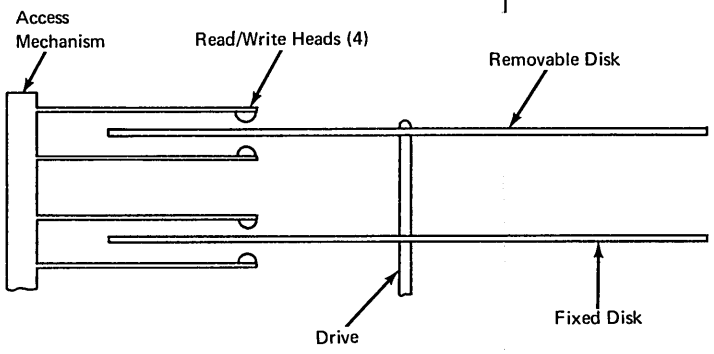
The disk storage drive reads data that is recorded on magnetic disks.



53978

The basic Model 6 installation includes one disk drive. The drive contains two disks and an access mechanism. One disk is fixed (F1); the other is removable (R1). The fixed disk (F1) cannot be physically removed. The removable disk (R1) can be replaced with another disk. (An optional feature of the Model 6 includes a second IBM 5444 Disk Storage Drive.)

The information on these disks can be replaced many many times and therefore used many more times than if you had to have a new disk every time you wanted to store more information.



ART: 52695

Tracks

Each disk is divided into circles called *tracks*. Depending upon the model of the IBM 5444 you have, you can record data on 200 to 400 tracks or 100 to 200 cylinders. Corresponding tracks from each side of the same disk are called *cylinders.* Each track is divided into 24 *sectors.* Each sector has its own unique address and can contain 256 characters of data. Six tracks (tracks 2-7) are used as alternate tracks. Tracks 0 and 1 are used only by the system.

Disk Organization

In order for your program to process data, you must store it somewhere on disk. Each piece of data (date, customer number, product number, etc.) is a *field.* Fields are grouped together to form a *record.* A *file* is a group of related records. There are five types of files for the Model 6:

Input Files. Input files are records that a program uses as a source of data. The program reads data from an input file and processes it.

Output Files. Output files are records written, punched, or printed by a program. The data in these has been processed.

Update Files. Update files are disk files from which a program reads a record, updates fields in the record, and writes the record back in the location from which it was read.

Combined Files. Combined files (ledger card files) are both input and output files. The program processes input data in this file and puts data that has been processed in the same file.

Display Files. A display file allows you to print the contents of up to two fields used in your program on the IBM 2265 Display Station.

Libraries

Not only can you store data on disk, but you can also store your programs on disk where they will be available for repeated use. The area on disk reserved for this is called a *library.* There are two libraries for your programs:

Source Library. The source library contains procedures and source statements.

Object Library. The object library contains object programs and routines. (When you indicate that you will include system programs in the object library, the system reserves space for a scheduler work area. The *scheduler work area* is a work area for one of the system programs, the Scheduler.)

PART II. CONVERSATIONAL OCL

Before the IBM System/3 Model 6 can run a program, it must know what you want it to do and where to find the information it will need to do the job. You supply the what and where information in a series of *OCL (operation control language) statements.* You must supply a series of OCL statements for every program you run.

Assume that you want to run an invoicing program. The program, which you have named INVOIC, requires three files: a customer master file and an inventory master file as input, and a tranaction file from which records of the ordered items are read. These three files are stored on disk. So that the program can be properly executed, you must supply the following information in a series of OCL statements:

WHAT is the name of the program?	INVOIC
WHAT is the date of this run?	12-06-71
WHAT files are used?	Customer master file, inventory master file, and transaction file.
WHERE is the program stored?	On the removable disk on drive one (R1).
WHERE are the files located?	Customer master file is on the fixed disk on drive one (F1). Inventory master file is also on the fixed disk on drive one (F1). Transaction file is on the removable disk on drive one (R1).

The OCL for the Model 6 is called *conversational OCL* because a question and answer procedure is used. The system prints the question called a *keyword,* and the operator supplies the answer called a *response.* The keyword tells the operator the type of information required by the system. For example, the keyword FILE NAME indicates that the name of one file used in the program must be supplied. By printing a keyword, the system is *prompting* the operator for a response.

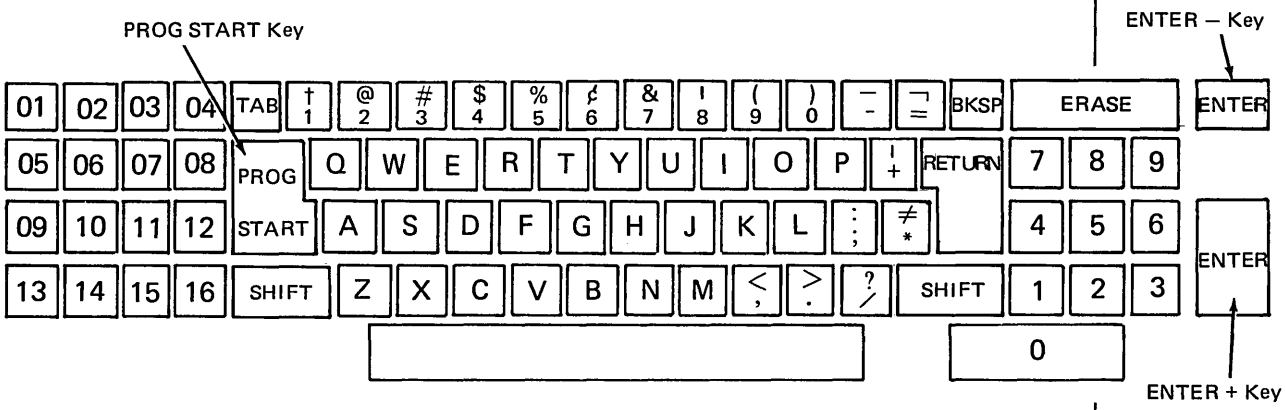
The operator responds to each keyword that applies to the job by typing in the relevant information. (When the system prompts FILE NAME, for example, the operator types the name of one file that the job uses.) If the system prompts a keyword that doesn't apply to the job, the operator bypasses the response.

REMEMBER. . .

- You must supply information in the form of OCL statements that the system needs to run the job.
- The IBM System/3 Model 6 uses conversational OCL which consists of keywords and responses.

The operator responds to a keyword by typing in a response (if the keyword applies to the job), and by pressing an *end-of-statement key.* Whether or not the operator types a response to a keyword, he must press an end-of-statement key before the system will prompt another keyword.

There are three end-of-statement keys on the Model 6 keyboard: ENTER-, ENTER+, and PROG START (Program Start).



Pressing the PROG START or ENTER+ key indicates the end of a response. Pressing the ENTER- key indicates the end of a response and may cause the system to skip several keywords. How many keywords are skipped depends on the keyword after which the ENTER- key was pressed. The PROG START and ENTER+ keys are interchangeable. As a matter of convenience, the PROG START key is the one usually specified in IBM's programming manuals for Model 6.

REMEMBER. . .

- Pressing an end-of-statement key completes an OCL statement.

If you want to test yourself on the material presented in chapters 2 and 3, see the self-test questions in Part V of this manual.

The system can't run any of your programs unless each one is accompanied by a series of OCL statements. A series of OCL statements is called an *OCL cycle.* There are four OCL cycles: LOAD, BUILD, BUILDC, and CALL.

Of the four cycles, only the LOAD cycle is independent; that is, you can run a job by responding just to the keywords in that cycle. The other three cycles are interrelated; to run a job you must use two or more of them.

The OCL cycle you choose to use should be based on frequency of program use and whether the program will be run alone or with a group of programs. The following chart may be used as a guide in this choice:

Type of Job	OCL Cycle
Jobs you run occasionally	LOAD
Jobs you run frequently	BUILD and CALL
Jobs you run together as a group *	BUILD, BUILDC, and CALL
* If it's a group of jobs you run frequently.	

THE LOAD CYCLE

When you use a LOAD cycle, you're telling the system:

1. Here are the OCL statements for my program.
2. Go to the disk drive I specify and find the program I want to run.
3. Load the program into the processing unit.
4. Run my program.

The LOAD cycle OCL statements are not saved. If you want to run the same job a second time, your operator must respond to all the keywords in the LOAD cycle again. It's best to use the LOAD cycle for jobs you run infrequently because this cycle has many keywords and takes quite a while for responses.

The following shows how you (the programmer), the operator, and the system would interact using the LOAD cycle:

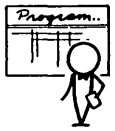
You, as the programmer, fill out an Operator's OCL Guide (see *Appendix A*) telling the operator to use the LOAD cycle for the job and how to respond to each keyword in the cycle.

Operator responds to all the keywords in the LOAD cycle.

System loads the LOAD cycle OCL statements in the processing unit and runs the job.

Will you ever want to run the job again?

YES



Save the old Operator's OCL Guide for the operator.

THE BUILD CYCLE

When you use a BUILD cycle, you're telling the system:

1. Here are the LOAD cycle OCL statements for job xxxx.
2. Store the LOAD cycle statements on disk so that they can be used whenever I want to run the program.
3. Do not run the program now.

Once the set of OCL statements is written on a disk, the set of statements is referred to as a *procedure.* The process of writing the statements on the disk is referred to as *building a procedure.* You use the BUILD cycle to build a procedure.

Although the BUILD cycle is the longest of all the OCL cycles in terms of operator time required, it doesn't run a job. Its function is to save the OCL statements for a job by writing them on one of the disks. The advantage of the BUILD cycle is that once the OCL statements are stored on disk, the program can be run using them rather than by keying all the required statements.

THE CALL CYCLE

CALL is the shortest OCL cycle, having only four keywords. When you use a CALL cycle, you're telling the system:

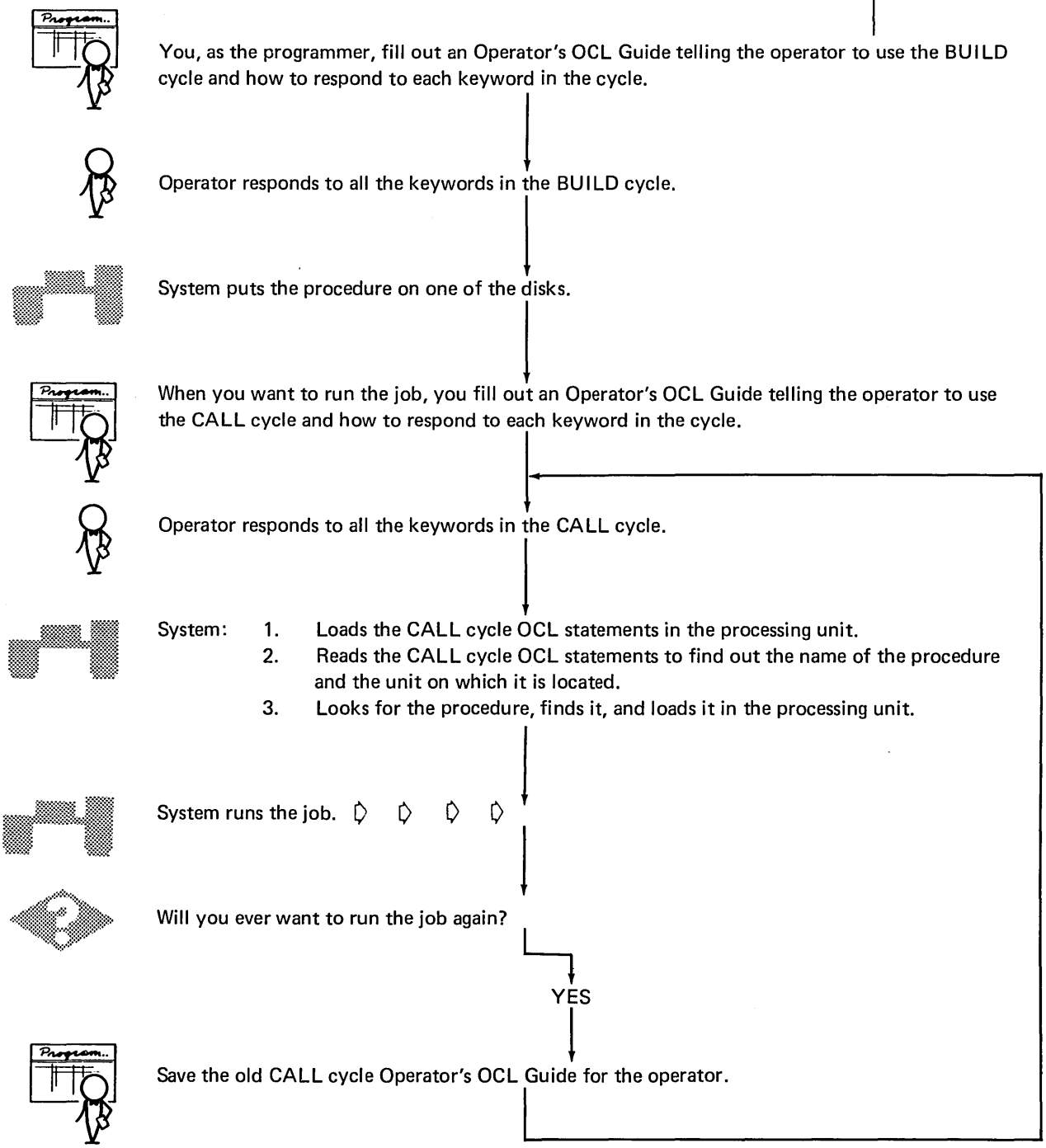
1. Locate, on disk, the procedure I built for job xxxx.

2. Use it to run job xxxx.

The CALL cycle is always linked to a BUILD or a BUILDC cycle.

THE INTERRELATIONSHIP OF THE BUILD AND CALL CYCLES

The following chart shows how you, the operator, and the system interact using the BUILD and CALL cycles:



The BUILD Cycle

After the operator finishes responding to the keywords in the BUILD cycle, the system writes the LOAD cycle OCL statements on disk. Remember that after the OCL statements have been written on the disk, they're referred to as a procedure.

The CALL Cycle

As the operator responds to each keyword in the CALL cycle, the system loads the statement into the processing unit.

The system then looks for the procedure identified by the CALL statements. When they system finds the procedure, it loads the OCL statements into the processing unit and runs the job.

THE BUILDC CYCLE

When you use a BUILDC cycle, you're telling the system:

1. I want to prepare a procedure to run a series of jobs which are always executed one after the other with no interruption.

2. The OCL statements for each job in the group are in procedures stored on disk.

3. Here are the names and disk drive locations of the procedures for each job in the group.

4. Build a chained procedure, establishing a sequence in which the individual procedures are run.

A *chained procedure* is a list of the procedures for each job in a group, in the order you want to run them. The list contains:

1. The name of the procedure for each job.

2. The disk drive on which the procedure is located.

The process of writing the list on a disk is referred to as *building a chained procedure.* BUILDC stands for build chained.

For example, here's the type of information you might find in a chained procedure for a weekly inventory job which consists of three separate jobs:

procedure name (for the 1st job in the group) — xxxx

procedure location (for the 1st job in the group) — (R1, R2, F1, or F2)

procedure name (for the 2nd job in the group) — yyyy

procedure location (for the 2nd job in the group) — (R1, R2, F1, or F2)

procedure name (for the 3rd job in the group) — zzzz

procedure location (for the 3rd job in the group) — (R1, R2, F1, or F2)

When you want to run the group of jobs, you use a CALL cycle to tell the system what chained procedure to use and where it's located.

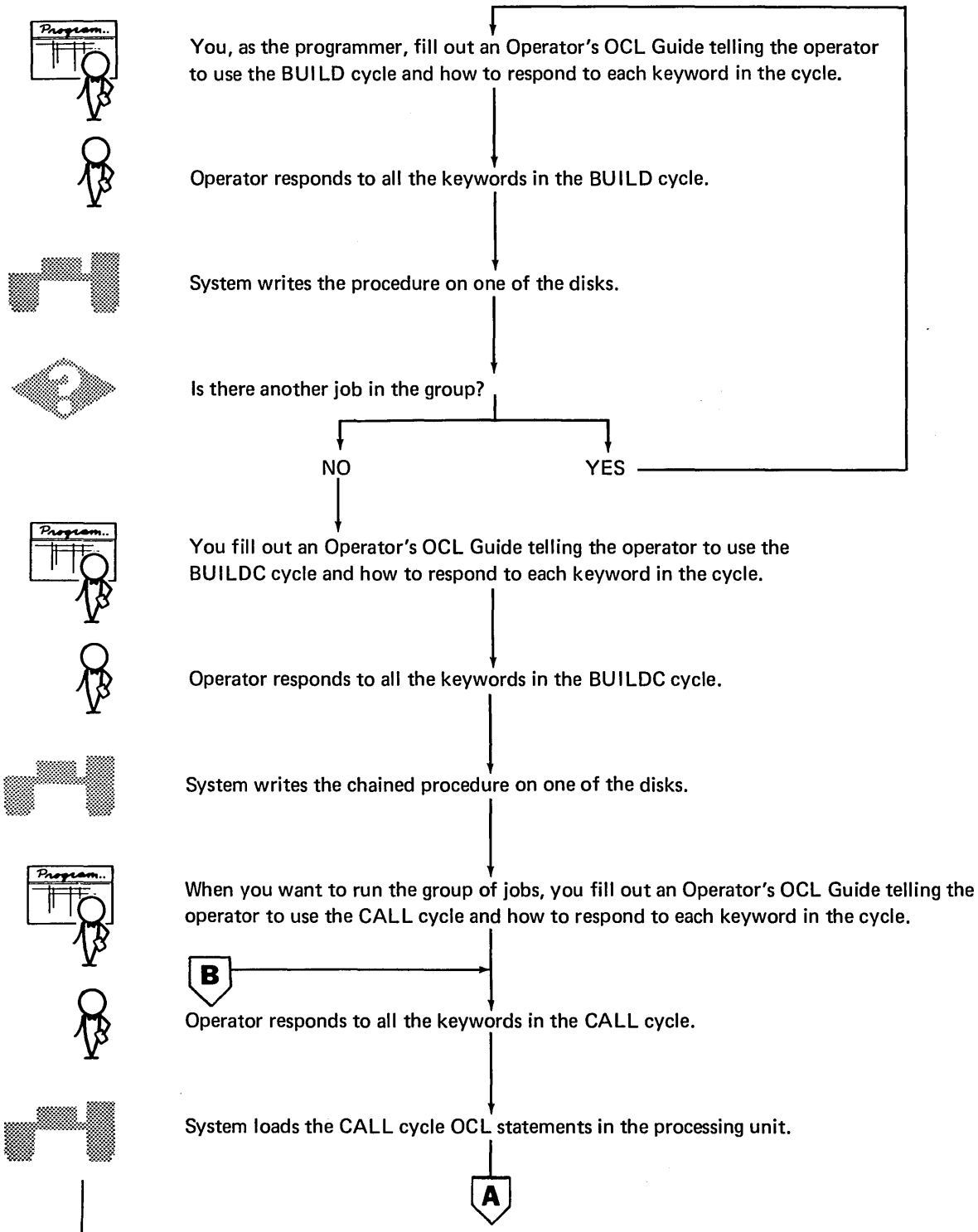
The advantage in using a BUILDC cycle for a group of jobs is that it allows you to run the jobs without stopping between each one to supply OCL statements. The disadvantage is that you must use three different cycles to set up and run one job. First, you must use the BUILD cycle to put the OCL statements for each job into a procedure. You use the BUILD cycle for each job in the group.

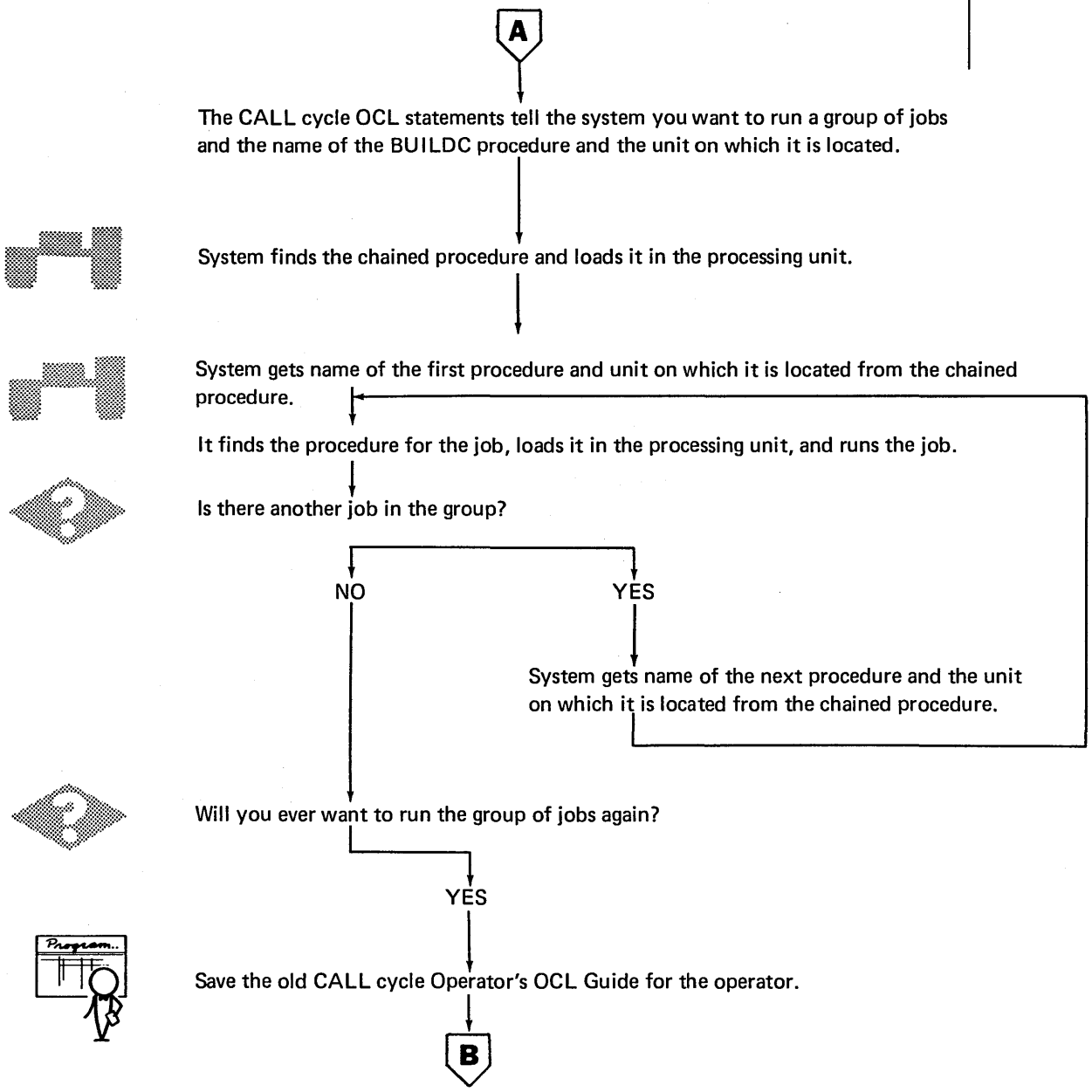
Next you use the BUILDC cycle to build a chained procedure which will contain the name of the procedure and the unit on which it is located for each job in the group. When you want to run the group of jobs, you must use a CALL cycle to tell the system what chained procedure to use and where it's located.

When you have a group of jobs you want to run together frequently, the BUILDC cycle can save you time. The amount of time spent on the BUILD and BUILDC cycles is compensated for by the time saved by being able to run a series of jobs with one CALL cycle. If you run the group of jobs only occasionally, using a series of LOAD cycles would be simpler and faster.

THE INTERRELATIONSHIP OF THE BUILD, BUILDC, AND CALL CYCLES

The following chart shows how you, the operator, and the system interact using the BUILD, BUILDC, and CALL cycles:




A
The CALL cycle OCL statements tell the system you want to run a group of jobs and the name of the BUILDC procedure and the unit on which it is located.
System finds the chained procedure and loads it in the processing unit.
System gets name of the first procedure and unit on which it is located from the chained procedure.
It finds the procedure for the job, loads it in the processing unit, and runs the job.
Is there another job in the group?
NO
YES
System gets name of the next procedure and the unit on which it is located from the chained procedure.
Will you ever want to run the group of jobs again?
YES
Program...
Save the old CALL cycle Operator's OCL Guide for the operator.
B

The BUILD Cycle

After the operator finishes responding to the keywords in the BUILD cycle, the system writes the OCL statements on disk. You use one BUILD cycle for each job in the group.

The BUILDC Cycle

You use the BUILDC cycle to build a chained procedure which consists of the name of the procedure and the unit on which it is located for each job in the group. Procedures are run in the order you enter the procedure information in the chained procedure.

The CALL Cycle

As the operator responds to each keyword in the CALL cycle, the system loads the CALL cycle statement into the processing unit. The statements tell the system:

- You want to run a group of jobs.
- The name of BUILDC procedure and the disk unit on which it is located.

The system looks for the BUILDC procedure, finds it, and loads it into the processing unit. The BUILDC procedure tells the system the name and disk unit of the first procedure to be run. The system then finds that procedure, loads it into the processing unit, and runs the job.

When the first job is complete, the system goes back to the chained procedure to see if there's another job in the group. If there is, the system finds the procedure for that job, loads it into the processing unit, and runs the job. This continues until every job in the group has been run.

REMEMBER. . .

- There are four OCL cycles: LOAD, BUILD, BUILDC, and CALL.
- You should choose the cycle based on frequency of program use and whether the program will be run alone or with a group of programs.

If you want to test yourself on the material in this chapter, see the self-test questions in Part V of this manual.

The first keyword in every OCL cycle is *READY.* When the system is ready to start a new job, it prompts the keyword READY, and the operator responds with the name of the OCL cycle you want to use for the job.

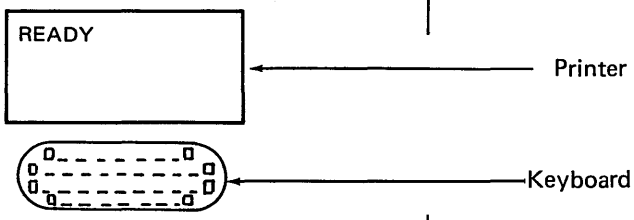
A function of the READY statement is to tell the system which OCL cycle you want to use for the job:

- READY-LOAD tells the system to prompt the keywords in the LOAD cycle.
- READY-BUILD tells the system to prompt the keywords in the BUILD cycle.
- READY-BUILDC tells the system to prompt the keywords in the BUILDC cycle.
- READY-CALL tells the system to prompt the keywords in the CALL cycle.

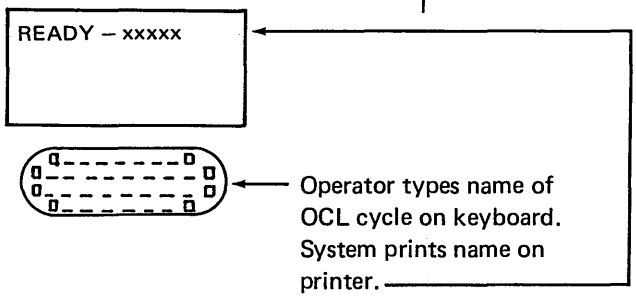
The READY statement can also be used to assign either the cathode ray tube or printer as the logging device. This is done by a response of LOG.

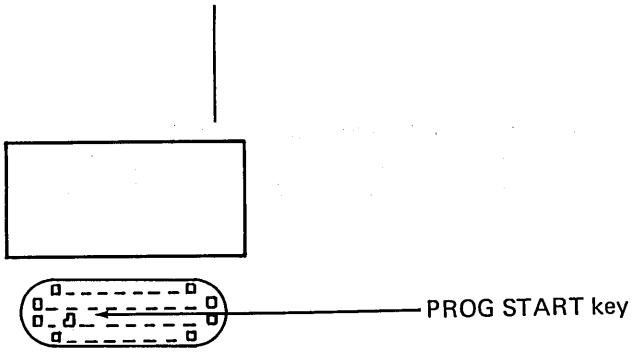
Here is what happens every time you start a new job.

1. When the system is ready to start a new job, it prompts READY.

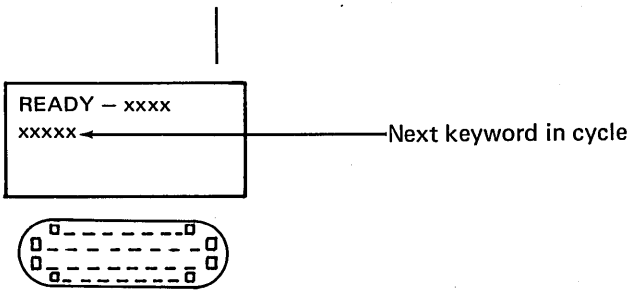


2. The operator responds to the keyword READY by typing the name of the OCL cycle you want to use for the job.





3. After typing in the name of the OCL cycle you want to use for the job, the operator presses the PROG START key.



4. As soon as the operator presses the PROG START key, the system prompts the next keyword in the cycle.

REMEMBER. . .

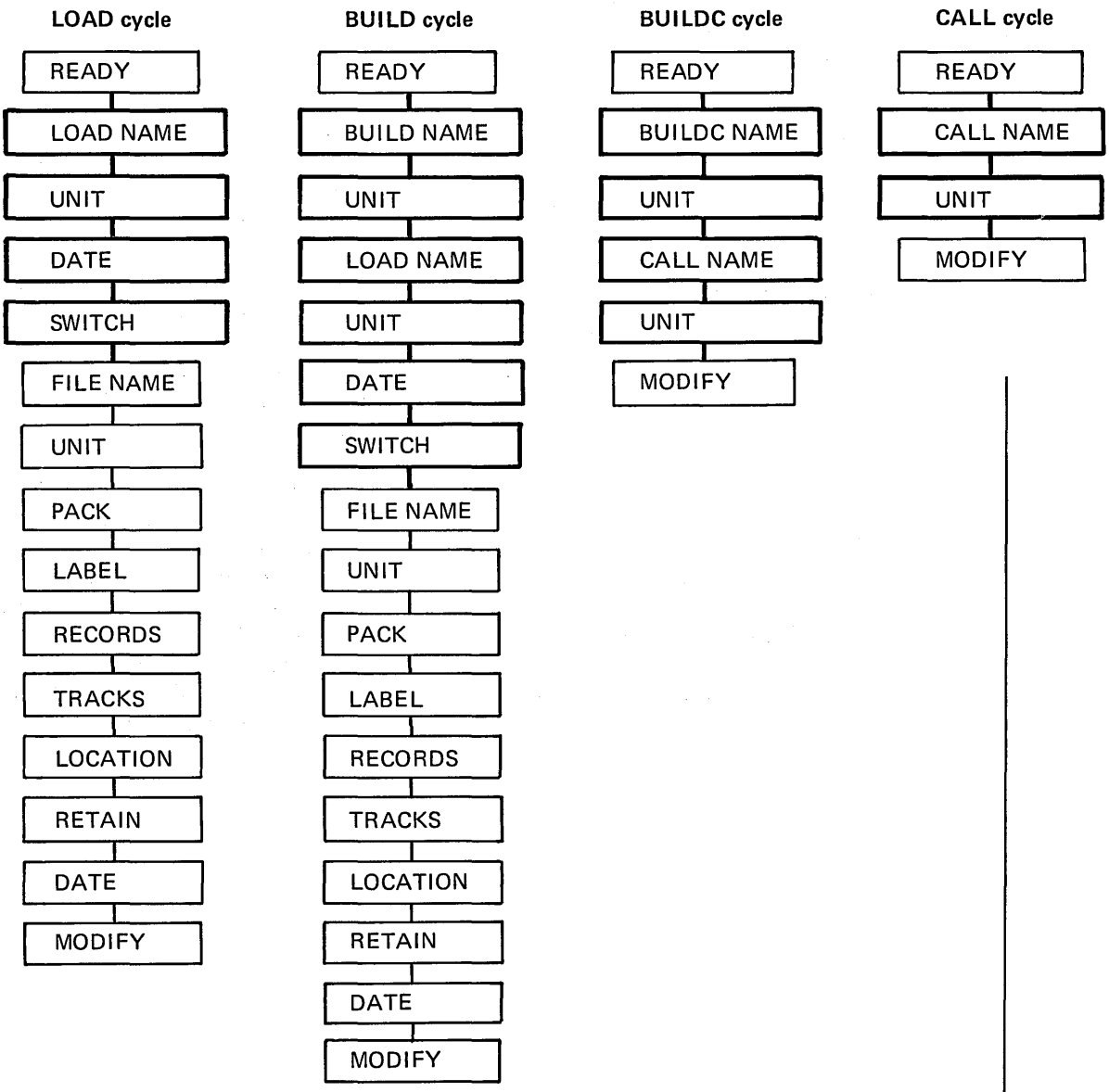
The READY statement tells the system:

- What OCL cycle you want to use for your job.
- You want to change the logging device.

CHAPTER 6. THE PROGRAM KEYWORDS

Each OCL cycle has a group of keywords referred to as the *program keywords.* When the system prompts one of the program keywords, it's asking for some particular information about the program you want to run. The operator's response to the keyword gives the system this information.

The program keywords are UNIT, DATE, SWITCH, and the four name keywords: LOAD NAME, BUILD NAME, BUILDC NAME, and CALL NAME. Not every cycle prompts each program keyword. DATE and SWITCH, for example, are prompted only during the LOAD and BUILD cycles. Here's how the program keywords fit into each cycle.



LOAD CYCLE

LOAD NAME

When the system prompts LOAD NAME, it's asking for the name of the program you want to load into the processing unit.

UNIT After LOAD NAME

When the system prompts UNIT after LOAD NAME, it's asking for the unit on which this program is located. There are four possible responses to UNIT:

- R1 – The removable disk on the first disk drive.
- R2 – The removable disk on the second disk drive.
- F1 – The fixed disk on the first disk drive.
- F2 – The fixed disk on the second disk drive.

DATE

When the system prompts DATE, it's asking what date you want to use for your job. Your response determines what date goes on the printed output for the job. This date is also used as the file date for any files created by running this job.

If you want to use the system date for your job, the operator should respond to DATE by pressing the PROG START key. (The system date is always established at IPL time.)

If you don't want to use the system date, the operator should respond to DATE by typing in a new date before pressing the PROG START key. This new date changes the system date for the one job only. When your job is finished, the system date will automatically revert to its IPL setting.

SWITCH

When the system prompts SWITCH, it's asking whether you want to change the setting of the eight external indicators. (Only RPG II programs use external indicators.)

If you don't want to change the setting, or if the program you want to run doesn't use external indicators, the operator should respond to SWITCH by simply pressing the PROG START key.

If you do want to change the setting of the external indicators, the operator should respond by typing in a new setting before pressing the PROG START key. (The *IBM System/3 Model 6 Operation Control Language and Disk Utility Programs Reference Manual,* GC21-7516 gives more detailed information about responding to the program keywords.)

BUILD CYCLE

BUILD NAME

When the system prompts BUILD NAME, it's asking what you want to name the procedure you're building.

UNIT After BUILD NAME

When the sytem prompts UNIT after BUILD NAME, it's asking for the unit on which you want to put the procedure. See *LOAD Cycle* in this chapter for the possible responses to UNIT.

LOAD NAME, UNIT After LOAD NAME, DATE, and SWITCH

See *LOAD Cycle* in this chapter.

Note: Delayed responses are valid for the BUILD cycle program keywords: UNIT after LOAD NAME, DATE, and SWITCH. A delayed response causes the system to reprompt the keyword during the CALL cycle and forces the operator to respond.

BUILDC CYCLE

BUILDC NAME

When the system prompts BUILDC NAME, it's asking what you want to name the chained procedure you're building.

UNIT After BUILDC NAME

When the system prompts UNIT after BUILDC NAME, it's asking for the unit on which you want to put the chained procedure. See *LOAD Cycle* in this chapter for the possible responses to UNIT.

CALL NAME, UNIT After CALL NAME

See *CALL Cycle* in this chapter.

CALL CYCLE

CALL NAME

When the system prompts CALL NAME, it's asking for the name of the procedure you want to use to run your job.

UNIT After CALL NAME

When the system prompts UNIT after CALL NAME, it's asking for the unit on which the procedure (or chained procedure) you want to use to run your job is located. See *LOAD Cycle* in this chapter for the possible responses to UNIT.

USING END-OF-STATEMENT KEYS WITH THE PROGRAM KEYWORDS

The ENTER- key may be used after LOAD NAME and UNIT in both the LOAD and BUILD cycles and after CALL NAME and UNIT in the BUILDC cycle. ENTER- is used after LOAD NAME and UNIT in the LOAD cycle to prompt MODIFY. It is used if you don't want to use any files for a program. ENTER- is used after LOAD NAME and UNIT in the BUILD cycle to prompt the compile keywords. In the BUILDC cycle it is used after CALL NAME and UNIT when all the procedure names have been entered to prompt MODIFY. After the rest of the program keywords PROG START is the only valid response.

REMEMBER...

- Each OCL cycle has a group of program keywords specifying particular information about the program you want to run.
- The following chart lists the program keywords and what each is asking for:

Program Keyword	Asks
LOAD NAME	Name of the program you want to load into processing unit.
BUILD NAME	What you want to name the procedure you're building.
BUILDC NAME	What you want to name the chained procedure you're building.
CALL NAME	Name of the procedure (or chained procedure) you want to use to run your job.
UNIT	Disk drive (R1, R2, F1, or F2) on which the program (or procedure) prompted by the preceding keyword is located.
DATE	Date you want to use for your job.
SWITCH	Whether you want to change the setting of the external indicators.

If you want to test yourself on the material presented in chapters 5 and 6, see the self-test questions in Part V of this manual.

Compiling an RPG II source program is much like any other job you run on the Model 6. There is one difference: to compile one of your RPG II source programs, the system must know which program you want to compile, on which disk drive it is located, and where you want to put the object program after compilation. Since this information is not supplied by your responses to the regularly prompted keywords, the system gets the information by prompting the three *compile keywords:* COMPILE OBJECT, SOURCE, and UNIT.

Your responses to the compile keywords give the system the information it needs to compile your RPG II source program:

Compile Keyword	Asks
COMPILE OBJECT	Where you want the system to write the object program after it has been compiled
SOURCE	Name of the RPG II source program you want to compile
UNIT	Disk drive on which the RPG II source program you want to compile is located

You can use either a LOAD or CALL cycle to compile an RPG II program. Using the CALL cycle is faster and easier because much of the information the system needs is already on the disk. (This is the only time you can use a CALL cycle to run a job without first having to use a BUILD cycle to put a procedure on a disk. This is because a procedure to run the RPG II Compiler program is on the system disk when you receive your Model 6.)

If you use the CALL cycle to compile your RPG II source program by running the IBM-supplied procedure named RPG, the system will pause while printing the procedure to prompt the three compile keywords:

	Keyword	Response	Comments
	READY	CALL	
	CALL NAME	RPG	Name of the IBM-supplied procedure to run the RPG II Compiler
	UNIT	XX	Disk drive (R1, R2, F1, or F2) on which the procedure is stored
	System prints out first part of procedure		
Compile Keywords	COMPILE OBJECT	XX	Disk drive (R1, R2, F1, or F2) on which you want the system to write the object program after compilation
Compile Keywords	SOURCE	YYYYY	Name of the RPG II program you want to compile
Compile Keywords	UNIT	XX	Disk drive (R1, R2, F1, or F2) on which the RPG II source program is stored
	System prints out remainder of procedure		
	MODIFY	RUN	

If you use the LOAD cycle, your response to LOAD NAME tells the system:

- I want to compile an RPG II program.
- Interrupt the normal LOAD cycle to prompt the three compile keywords.

	Keyword	Response	Comments
	READY	LOAD	
	LOAD NAME	$RPG	This response tells the system you want to run the RPG II Compiler
	UNIT	XX	Disk drive (R1, R2, F1, or F2) on which the RPG II Compiler is located
Compile Keywords	COMPILE OBJECT	XX	Disk drive (R1, R2, F1, or F2) on which you want the system to write the object program after compilation
	SOURCE	YYYYY	Name of the RPG II program you want to compile
	UNIT	XX	Disk drive (R1, R2, F1, or F2) on which the RPG II source program is stored
	The system resumes the normal prompt-response sequence for the rest of the LOAD cycle.		

(See the *IBM System/3 Model 6 Operation Control Language and Disk Utility Programs Reference Manual,* GC21-7516 for more detailed information on compiling RPG II programs.)

If you use the BUILD cycle, you can respond to the compile keywords with a delayed response. A *delayed response* causes the system to reprompt the keyword during the CALL cycle and forces the operator to respond.

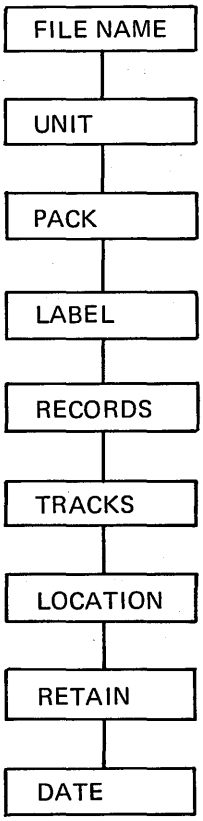
REMEMBER...

- If you're writing your programs in the RPG II programming language, you'll use the RPG II Compiler to translate your programs into machine language.
- To compile an RPG II program, the system prompts the three compile keywords: COMPILE OBJECT, SOURCE, and UNIT.
- You can use either a LOAD or CALL cycle to run the RPG II Compiler.

CHAPTER 8. THE FILE KEYWORDS

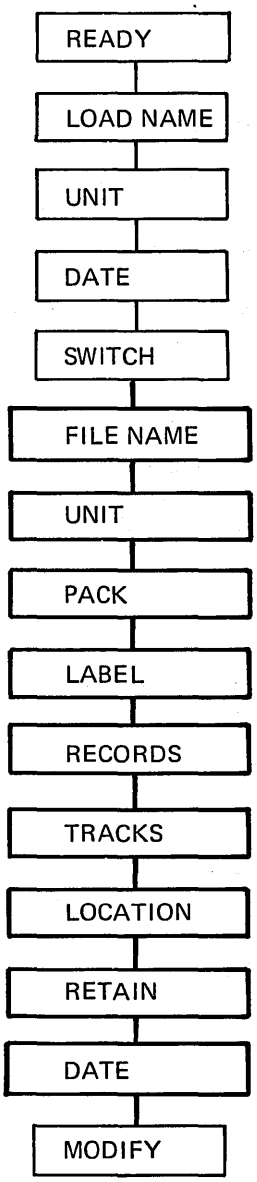
To get information about the files that are going to be used in a job, Model 6 prompts a series of keywords called the *file keywords.* When the system prompts a file keyword, it's asking for some specific information about one of the files used in your job.

The file keywords are always prompted in the sequence shown and are only prompted during LOAD and BUILD cycles.

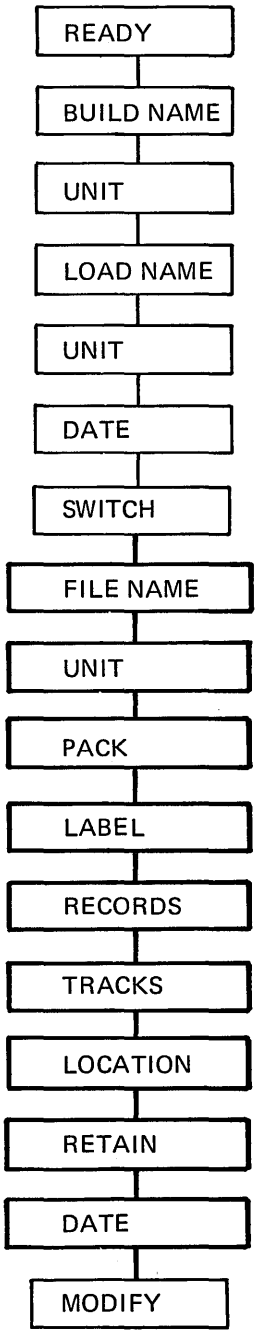


Here's how the file keywords fit into the two cycles:

LOAD cycle



BUILD cycle



For every file a job uses, the operator must respond to the series of file keywords. If a job uses several files, 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. The second time the system prompts the file keywords, the operator responds with information about a second file. The system continues to prompt the series of file keywords until the operator has described all the files the job uses.

RESPONDING TO THE FILE KEYWORDS

When the system prompts a file keyword, it's asking for specific information about one of the files used in your job. For every file a job uses, you must provide a response for the first three file keywords: FILE NAME, UNIT, and PACK.

FILE NAME

FILE NAME asks for the name of one file that the job uses. For a file used in an RPG II customer program, the response to FILE NAME is the name in columns 7-14 of the RPG II File Description Specifications Sheet. Also, a predefined file name is used for certain Model 6 programs. (The *IBM System/3 Model 6 Operation Control Language and Disk Utility Programs Reference Manual,* GC21-7516 lists these programs.)

UNIT

UNIT asks for the disk drive (R1, R2, F1, or F2) containing the file.

PACK

PACK asks for the name of the disk containing the file. The name of the disk is the name assigned by the user to the pack when the pack was initialized.

LABEL

LABEL asks for the name by which the file can be identified when it is stored on disk. You respond to this prompt as follows:

1. If the identifying name and the previous response to FILE NAME are the same, no response is required. The operator presses PROG START. The system will assume that the two entries should be the same.

2. If the disk file identifying name and the previous response to FILE NAME aren't the same, the operator types the identifying name and then presses PROG START.

RECORDS and TRACKS

RECORDS and TRACKS are the two *space keywords.* When you're writing a file on disk for the first time, you must supply a response for one of the space keywords. (Writing a file on disk for the first time is often referred to as *creating a file.*) When the system prompts these keywords, it's asking how much space the file you're creating will take up on disk. If you're creating a file and you don't supply a response for either of the space keywords, the system won't be able to write your file on a disk because it doesn't know how much space is required.

RECORDS asks how many records are in the file you're describing. TRACKS asks how many disk tracks it takes to contain the records in the file. If you don't want to calculate how many tracks the records in your file will take up, respond to RECORDS; the system will do the records-to-tracks conversion for you. If you want to do the conversion yourself, rules for converting records-to-tracks for different types of files are given in the *IBM System/3 Model 6 Operation Control Language and Disk Utility Programs Reference Manual,* GC21-7516. After you've calculated the conversion, you can respond to TRACKS.

Note: You should allow room for future expansion of the file in your response if the file will be added to later.

LOCATION

LOCATION asks for a track number where you'd like your file to start. You supply a response to LOCATION only if you want to have complete control over the arrangement of the files on the disk. If you don't respond to LOCATION, the system will determine where the file is to be stored based on available disk space.

RETAIN

RETAIN asks for the file's designation: P, T, S, or A.

- P designates a permanent file. A permanent file is one which is expected to be maintained permanently on disk.
- T designates a temporary file. A temporary file is one which has short term usefulness and may be overwritten when this usefulness has ended.
- S designates a scratch file. A scratch file is intended for use only by the current program and may be overwritten by the next program. S is also used to change the designation of a temporary file so that its space will be available to subsequent programs.
- A designates an activated file (a file whose designation is being changed from S to T).

You must supply a response to RETAIN at certain times:

- At file creation time, if you want the file to be designated P or S.
- During a program run, if you want to change a file's designation.

If you don't respond to RETAIN during a file creation run, the file will have a T designation.

DATE

DATE asks for the date when an input or update file was created. This date is stored with the identification information for the file on disk, and is the same as the system date that was in effect when the file was created. The only time you must supply a response to DATE is when you're running a job which might have as input one of two or more files stored on the same disk pack whose identifying names (LABEL) are the same. In this case, the only way the system can determine which file to use is by verification of the file creation date. If no date is specified and two or more files exist with the same file name, the file with the latest date will be chosen. (The *IBM System/3 Model 6 Operation Control Language and Disk Utility Programs Reference Manual,* GC21-7516 discusses the file keywords in detail.)

KEYWORDS FOR MULTI-VOLUME FILES

If you have a file that can't be contained on one disk, you may continue it on one or more subsequent disks. This type of file is called a *multivolume file.* There are certain additional considerations in responding to a file keyword for multivolume files. (See the *IBM System/3 Model 6 Operation Control Language and Disk Utility Programs Reference Manual,* GC21-7516 for additional information on multivolume files.)

List Requirements

Some of the FILE statement parameters require lists when used for multivolume files. A list consists of a single quote, responses to the parameter separated by commas, and another single quote:

UNIT - 'R1,R2'

The PACK parameter always requires a list while UNIT may require a list. LOCATION, TRACKS, HIKEY, and RECORDS require a list if they are stated.

FILE NAME

See *FILE NAME* in this chapter for single volume files.

KEY LENGTH

If the operator presses the ENTER-key after responding with a file name to FILE NAME, an indexed multivolume file has been indicated. ENTER- prompts the file keyword KEY LENGTH, which asks for the length of the key field. If you respond to KEY LENGTH, another keyword (HIKEY) for indexed files is prompted. If you press PROG START after KEY LENGTH, HIKEY is bypassed.

HIKEY

HIKEY asks for the highest key field for a volume. You must respond to the HIKEY parameter for each volume, and that response (which specifies length) must equal the response to KEY LENGTH. The keyword applies to indexed multivolume files only.

UNIT

UNIT asks for the disk drives (R1, R2, F1, or F2) which contain the file. For indexed or consecutive multi-volume files, an entry can correspond to more than one disk name in the PACK statement. Assume the following responses are made to the UNIT and PACK keywords:

UNIT – 'R1,R2'

PACK – '1,2,3,4'

Processing of disks 1 and 3 will be on R1 and processing of disks 2 and 4 will be on R2. However, for direct files there must be a one-to-one correspondence between UNIT and PACK.

PACK

PACK asks for the names of the disks that contain the file. The disk names are the names you assigned to the pack. The number of PACK responses must correspond to the number of HIKEY responses (if used).

LABEL

See *LABEL* in this chapter for single volume files.

RECORDS and TRACKS

You must supply a response to one of these keywords. When prompted, RECORDS asks for the number of records in the file. TRACKS asks for the number of disk tracks it takes to contain the records in the file. The order of numbers in the response must correspond to the order of the names in the PACK parameter.

LOCATION

LOCATION asks for the number of the tracks where the file is to begin for each file on the disk. The order of the numbers must correspond to the order of the names in the PACK parameter. If you omit this keyword, the system will allocate space on each disk.

RETAIN

See *RETAIN* in this chapter for single volume files.

DATE

See *DATE* in this chapter for single volume files.

USING END-OF-STATEMENT KEYS WITH THE FILE KEYWORDS

Following your response to the first two file keywords (FILE NAME and UNIT), the only valid end-of-statement key for single volume files is PROG START. An ENTER-response to FILE NAME prompts KEY LENGTH for indexed multivolume files. As a response to the rest of the file keywords, you can use either the PROG START or ENTER- key, depending on what you want the system to do. Pressing the PROG START key after your typed response tells the system to prompt the next keyword. Pressing the ENTER- key after your typed response tells the system to skip the rest of the file keywords and prompt FILE NAME again. If the operator doesn't type in a response but merely presses the PROG START key after FILE NAME, the system will skip all the file keywords and prompt MODIFY. This indicates that all files have been described and you are ready to run the job (see Chapter 8 for a discussion of MODIFY).

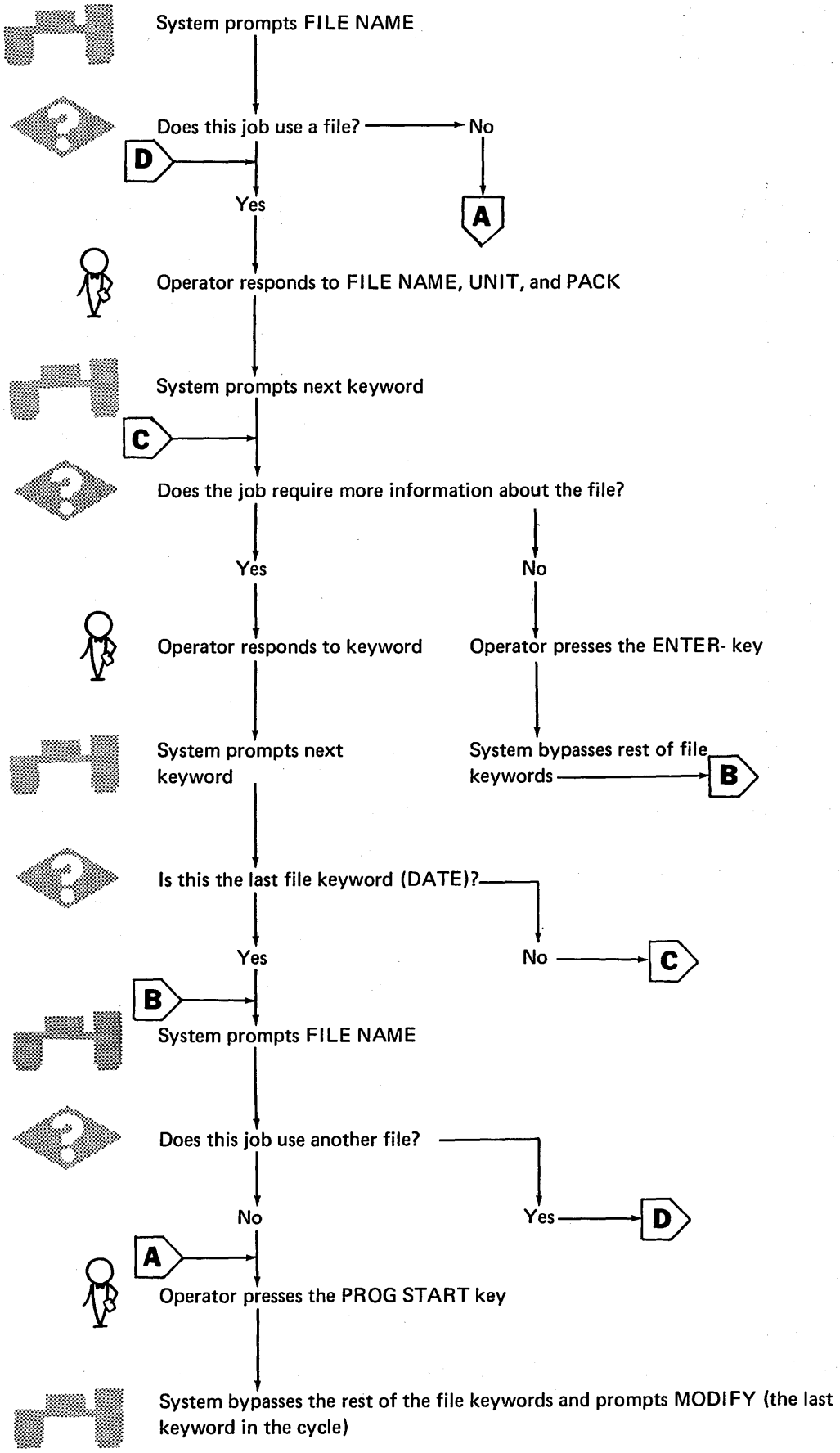
DELAYED RESPONSE

Responding to a keyword with a question mark is referred to as a *delayed response.* Delayed responses are only valid for the BUILD cycle, but can be used for all the file keywords in that cycle. Two things happen when a delayed response is given:

- The system reprompts the keyword during the CALL cycle.
- The operator is forced to respond to the keyword when it is reprompted. (The CALL cycle won't continue until the operator uses a valid response.)

REMEMBER. . .

- Your responses to the file keywords give the system information about the files you're using in your job.
- In a job situation you (the programmer), the operator, and the system interact in the following manner:


System prompts FILE NAME
Does this job use a file?
No
D
Yes
A
Operator responds to FILE NAME, UNIT, and PACK
System prompts next keyword
C
Does the job require more information about the file?
Yes
No
Operator responds to keyword
Operator presses the ENTER- key
System prompts next keyword
System bypasses rest of file keywords
B
Is this the last file keyword (DATE)?
Yes
No
C
B
System prompts FILE NAME
Does this job use another file?
No
Yes
D
A
Operator presses the PROG START key
System bypasses the rest of the file keywords and prompts MODIFY (the last keyword in the cycle)

- Some keywords always require a response, some require a response only when a certain type is being used, and some do not require a response.

File Keyword	Single *	Multivolume *
FILE NAME	▒	▒
KEY LENGTH **		▒
HIKEY **		▒
UNIT	▒	▒
PACK	▒	▒
LABEL		
RECORDS ***	▒	▒
TRACKS ***	▒	▒
LOCATION		▒
RETAIN		
DATE		

* Shaded blocks indicate which responses are required.

** Not required unless indexed multivolume.

*** You must respond to only one of these when you're creating a file.

- The possible responses to keywords for single and multivolume files are as follows:

File Keyword	Response*					
	Single			Multivolume		
	Press PROG START	Type a response, press PROG START	Type a response, press ENTER—	Press PROG START	Type a response, press PROG START	Type a response, press ENTER—
FILE NAME	▒	▒		▒	▒	▒
KEY LENGTH **				▒	▒	
HIKEY **					▒	
UNIT		▒			▒	
PACK		▒	▒		▒	▒
LABEL		▒	▒		▒	▒
RECORDS		▒	▒		▒	▒
TRACKS		▒	▒		▒	▒
LOCATION		▒	▒		▒	▒
RETAIN		▒	▒		▒	▒
DATE		▒	▒		▒	▒

* Shaded blocks indicate which responses can be made to a file keyword.

** Indexed multivolume.

▒

If you want to test yourself on the material presented in chapters 7 and 8, see the self-test questions in Part V of this manual.

Not only is *MODIFY* the last keyword in every OCL cycle — it is also the most versatile. With a MODIFY statement you can:

- Run a job.
- Cancel a job.
- Correct one or more OCL statements in a cycle.
- Delete one or more OCL statements in a cycle.
- Enter LOG and FORMS statements to a cycle.
- Insert comments in a cycle.
- Include instructions for one of the system programs in a cycle.

RUNNING A JOB

When the operator is sure the OCL cycle is complete and correct, he should type RUN in response to the keyword MODIFY.

```
***************************
MODIFY
RUN
```

Tells the system to run the job.

CANCELING A JOB

To cancel a job after MODIFY, the operator types CANCEL and presses the space bar and the PROG START key. This tells the system to cancel the job and start prompting keywords for the next job. (The operator can respond with /* at any time to get an immediate end of job, instead of waiting for MODIFY.)

```
*************************
MODIFY
CANCEL
READY
```

Tells system to cancel job.

System starts prompting keywords for the next job.

CORRECTING AND DELETING OCL STATEMENTS

Not every statement in every cycle may be corrected or deleted. To show which statements may be corrected or deleted, the system outlines them with a border of asterisks.

```
READY                -LOAD

*******************************

001    STATEMENT
002    STATEMENT
003    STATEMENT

*******************************

MODIFY
```

These statements may be corrected or deleted.

After the system prompts MODIFY, the operator can correct or delete any of the statements within the border of asterisks. He does this by typing the statement number of the statement he wants to correct. The statement number is the 3-digit number to the left of each statement inside the border of asterisks.

```
READY                        -LOAD

***********************************

010   LOAD NAME              -PAYROL
011        UNIT              -R1
020   DATE (12/02/71)        -
030   SWITCH (11111111)      -
040   FILE NAME              -

***********************************

MODIFY

011   (PROG START)           -R2
```

Tells the system the operator is going to work with statement 011.

Tells the system to replace R1 with R2 in statement 011.

Correcting an OCL Statement

When the operator sees a mistake in one of the statements within the asterisk border, he can use a MODIFY statement to correct it. He waits until the system prompts MODIFY, types the 3-digit number of the incorrect statement, then presses the PROG START key. Pressing the PROG START key moves the printer to the response column where the operator can type the response he wants.

Deleting an OCL Statement

To delete a statement within the asterisk border, the operator responds to MODIFY by typing the statement number of the statement he wants deleted. He then types a comma and presses the PROG START key. A comma immediately following the statement number tells the system to remove that statement from the OCL cycle.

```
READY                                  -LOAD

*********************************

010    LOAD NAME                       -PAYROL
011         UNIT                       -R1
020    DATE (12/02/71)                 -01/10/72
030    SWITCH (11111111)               -
040    FILE NAME                       -

*********************************

MODIFY

020, (PROG START)
```

Tells the system to delete statement 020.

ENTERING LOG AND FORMS STATEMENTS

In every OCL statement we've talked about the system prompts with a keyword, and the operator types a response. There are two statements for which the operator types both the keyword and the response: the LOG and FORMS statements.

The LOG statement tells the system where to print OCL statements and error messages for a job: on the 13-inch printer, the primary tractor of the 22-inch printer, or the cathode ray tube. The FORMS statement tells the system how many lines to print on each page. (The FORMS statement doesn't apply to the cathode ray tube.)

The LOG and FORMS keywords are never prompted. The only way you can get the FORMS statement into an OCL cycle is to enter it after the system prompts MODIFY. LOG is a valid response to MODIFY or READY (see *Chapter 4. Beginning an OCL Cycle*). The *IBM System/3 Model 6 Operation Control Language and Disk Utility Programs Reference Manual,* GC21-7516 discusses the LOG and FORMS statements in detail.

INSERTING COMMENTS IN A CYCLE

Sometimes it is necessary to include statements in the OCL cycle which aren't instructions to the system. You may want to remind the operator to put a special kind of paper in the printer before running a job, or the operator may want to indicate the reason why he changed one of the statements in the cycle.

You and the operator use *comment statements* for this communication. Comment statements always stand out in an OCL cycle because they start with an asterisk (*).

The asterisk in front of the statement tells the system, "This is not an instruction for you." The system then ignores these statements when the job is run even though they are part of the OCL statements for the job.

To enter a comment statement, the operator responds to the keyword MODIFY by typing an * followed by the comment.

Note: The operator doesn't have to wait for the system to prompt MODIFY before he enters a comment statement. Comment statements can be entered anywhere in the OCL cycle. The *IBM System/3 Model 6 Operation Control Language and Disk Utility Programs Reference Manual,* GC21-7516, contains complete instructions for entering comment statements earlier in the cycle.

Comment from Operator

The operator might enter a comment to explain to you why he modified one of your statements.

```
READY                          -LOAD

********************************

010   LOAD   NAME          -PAYROL
011          UNIT          -F1
020   DATE   (12/06/71)    -
030   SWITCH(01011111)     -
040   FILE   NAME          -EMPMAS
041          UNIT          -R2
042          PACK          -VOL06
050   FILE   NAME          -

********************************

MODIFY

041                            -R1

*  R2 DOWN 12/6.  VOL06 MOVED TO R1
```

Comment statement from operator explaining why statement 041 was changed.

Comment from Programmer to Operator

```
READY                          - CALL
000 CALL NAME                  - MPAY
001      UNIT                  - F1
***********************************
010 LOAD NAME                  - PAYROL
011      UNIT                  - R1
020 DATE (12/06/71)            -
030 SWITCH (01011111)          -
040 FILE NAME                  - EMPMAS
041      UNIT                  - F2
042      PACK                  - VOL06
050 FILE NAME                  -
***********************************
MODIFY
* PUT CHRISTMAS PAPER ON PRINTER
```

Comment statement from you to operator with special job instructions. This statement was inserted into the procedure at BUILD time.

You should use a comment when special instructions must be given to the operator. These special instructions would be entered during the BUILD cycle so that when the operator runs the job, using the CALL cycle the instructions appear in the OCL listing, reminding the operator of the special requirements for the job.

INCLUDING INSTRUCTIONS FOR ONE OF THE SYSTEM PROGRAMS

Most OCL cycles contain only OCL statements. The BUILD cycle, however, can also contain instructions for some of the system programs.

The Model 6 system programs available from IBM which can use instructions included during a BUILD cycle are the Disk Utility programs and the Disk Sort program.

When you have two programs you always run together, and one requires program control statements, you can save both system and operator time by including the instructions for the system program in the OCL cycle for that program. Information on writing the program instructions for the system programs is given in the referenced manuals.

System Program	Manuals
Disk Utility programs	Part III of this manual
	IBM System/3 Model 6 Operation Control Language and Disk Utility Programs Reference Manual, GC21-7516
Disk Sort program	*IBM System/3 Disk Sort Reference Manual,* SC21-7522

To include instructions for a system program in a BUILD cycle, the operator responds to the keyword MODIFY by typing INCLUDE. The MODIFY-INCLUDE statement tells the system the operator is entering instructions for a system program. All that's left for the operator to do is type in the instructions.

```
READY                          - BUILD
*******************************
                 ↑
   BUILD Cycle OCL Statements
                 ↓
*******************************
MODIFY
INCLUDE
*******************************
ENTER INCLUDED STATEMENTS
System instruction
System instruction
System instruction
      etc.
```

Including system instructions in a BUILD cycle.

Putting instructions for one of the system programs in a BUILD cycle is referred to as *including system instructions in a procedure.* The system instructions are then referred to as the *included statements.*

The keyword MODIFY is prompted twice during a BUILD cycle that includes system program instructions. The first MODIFY applies to the OCL statements; the second applies to the included statements (the system program instructions).

```
READY                    - BUILD

000 BUILD NAME           - INITRI

001       UNIT           - F1

*********************************

   OCL Statements for LOAD Cycle

*********************************

MODIFY                                  This MODIFY keyword applies
                                        to the OCL statements.
INCLUDE

*********************************

ENTER UTILITY CONTROL STATEMENTS

      System Instructions

RUN

*********************************

MODIFY                                  This MODIFY keyword applies
                                        to the system instructions.
```

When you use the CALL cycle to run the two programs, the system prompts MODIFY twice. The first MODIFY applies to the OCL statements in the procedure; the second applies to the included statements.

SEVERAL MODIFY STATEMENTS IN ONE JOB

You may want to use several MODIFY statements in one job. Suppose, for example, that the operator completes the OCL statements for a BUILD cycle, checks the printed OCL statements against your OCL instructions, and finds that:

1. Three of the statements have mistakes.

2. One statement must be deleted entirely.

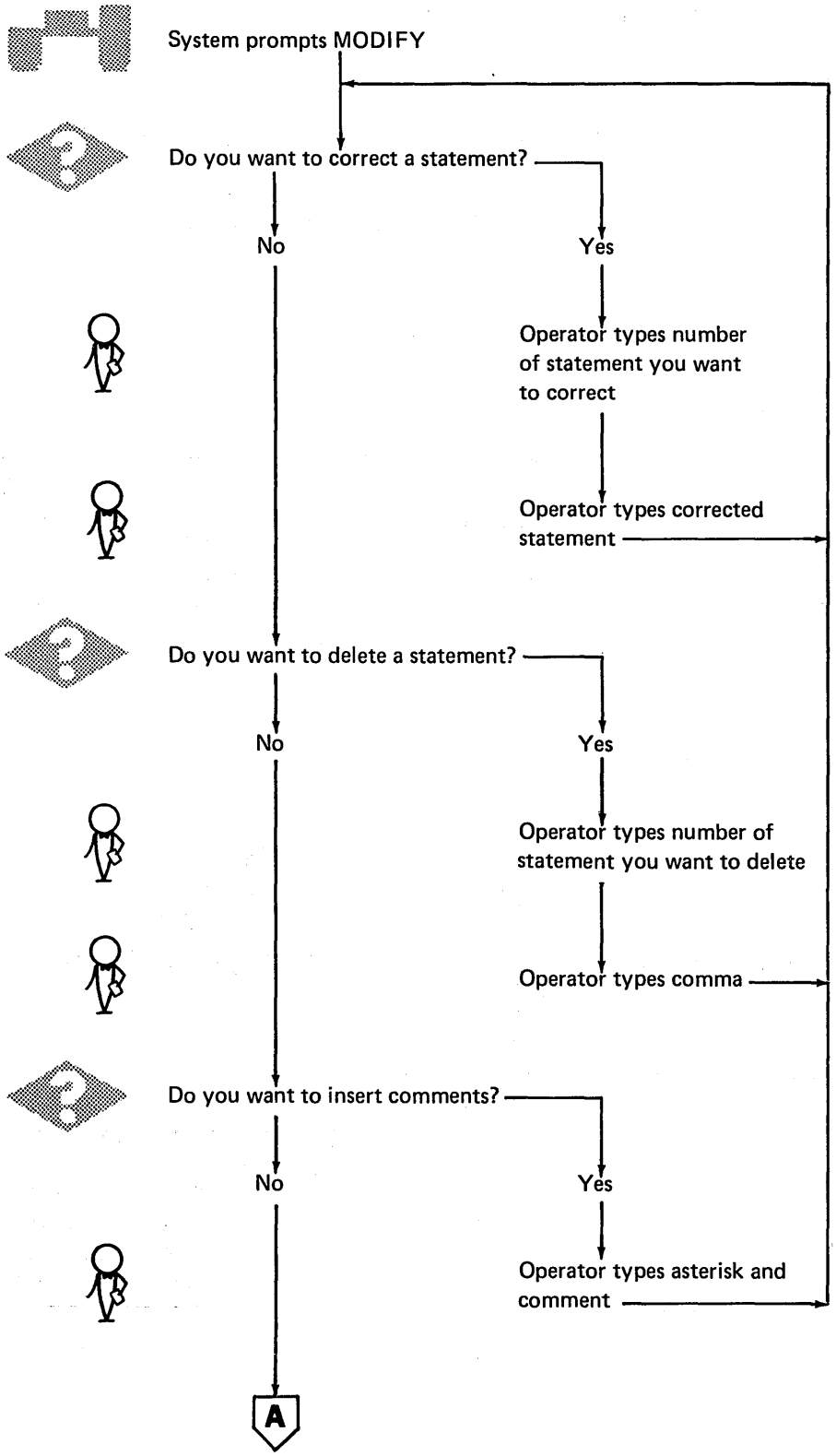
3. There are a set of instructions for the Disk Sort program that you want him to include in the procedure.

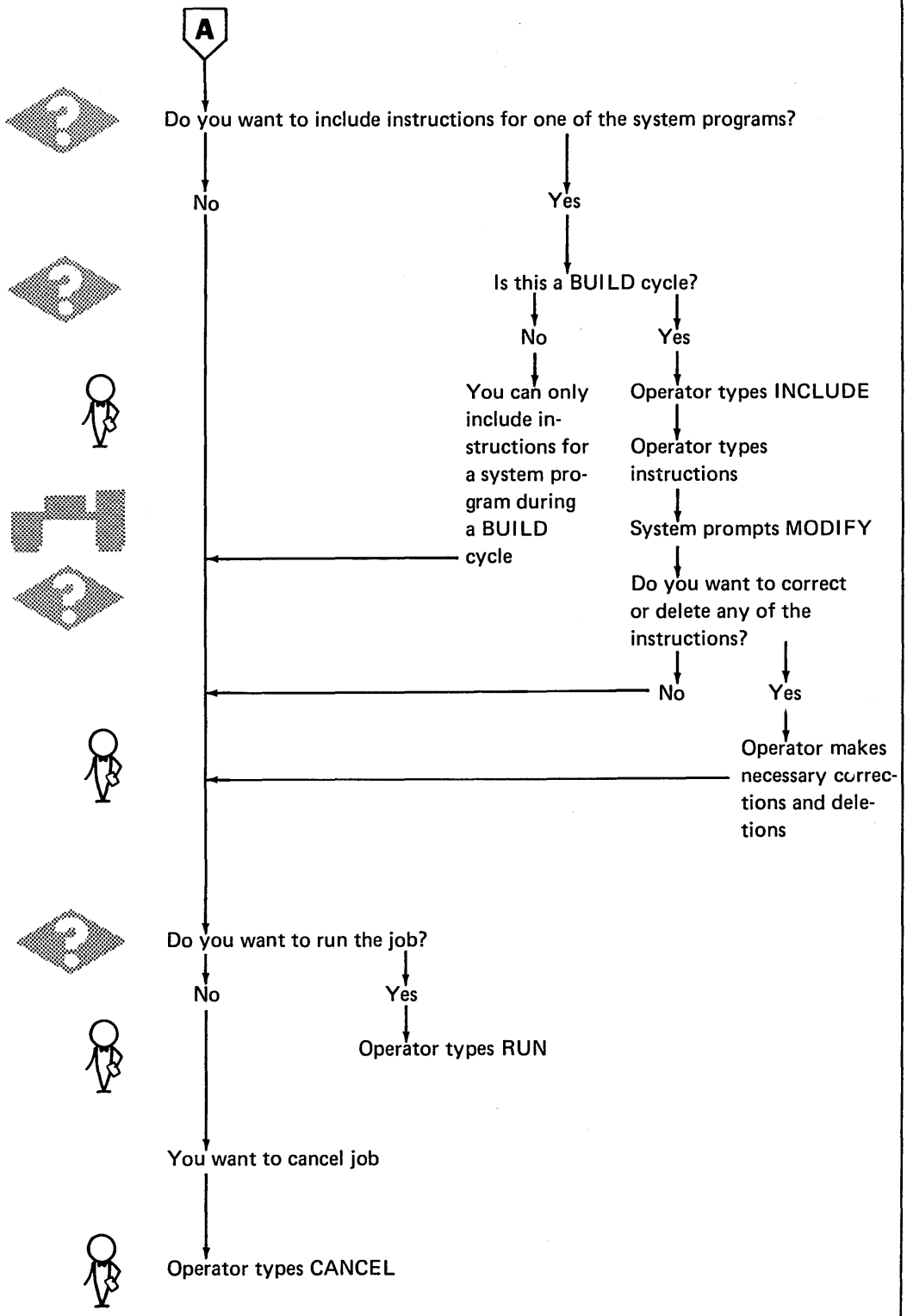
This is how the operator might handle the MODIFY part of the cycle:

```
******************************

MODIFY

010                 - PAYROL      Correction.  The previous statement
                                  was 010 BUILD NAME - PAYROX.
030                 - 01101100    Correction.  The previous statement
                                  was 030 SWITCH - 01100100.
041                 - R1          Correction.  The previous statement
                                  was 041 UNIT - R2.
060,                              Deletion.  The statement doesn't
                                  belong in this job.

INCLUDE

******************************

ENTER INCLUDED STATEMENTS

Instructions for Disk Sort Program      Included statements.

******************************

MODIFY              - RUN         This statement tells the system two
                                  things:
                                  1.  The included statements are
                                      complete and correct.
                                  2.  Now put the procedure on the
                                      disk so it's there when I want
                                      to use it.
```

REMEMBER. . .

- In a job situation using the MODIFY statement, you (the programmer), the operator, and the system interact in the following manner:





If you want to test yourself on the material presented in this chapter, see the self-test questions in Part V of this manual.

CHAPTER 10. ENDING THE OCL CYCLE

When the operator is sure the OCL cycle is complete and correct, he types RUN in response to the keyword MODIFY. The MODIFY-RUN statement tells the system to run the job.

As soon as the operator types RUN and presses the PROG START key, the system starts running the job. Running the job, however, means something different for each OCL cycle.

In an LOAD cycle, the MODIFY-RUN statement tells the system to run the program you specified in the LOAD NAME statement.

In a BUILD cycle, the MODIFY-RUN statement tells the system:

1. Write all the OCL statements in the cycle on the disk specified in the UNIT statement.

2. Get the procedure name from the BUILD NAME statement.

Remember that once the OCL statements for a job are written on a disk, the entire set of statements is referred to as a procedure.

In a BUILDC cycle, the MODIFY-RUN statement tells the system:

1. Write the name of the procedure and the unit on which it is located for every job in the group on the disk specified in the UNIT statement.

2. Get the chained procedure name from the BUILDC NAME statement.

   Remember that once the name of the procedure and unit on which that procedure is located for each job in the group are written on the disk they are referred to as a chained procedure because the system uses this information to chain (connect) the OCL statements for all the jobs in this group.

In a CALL cycle, the MODIFY-RUN statement tells the system:

1. Run the job specified in the CALL NAME statement.

2. The job is located on the disk drive specified in the UNIT statement.

The CALL cycle will have two MODIFY-RUN statements when the procedure you're calling contains included statements in addition to its OCL statements. For example, if you've built a procedure to sort your updated payroll master file and have included instructions for the Disk Sort program, your CALL cycle might look like this:

```
READY          - CALL

CALL NAME      - Name of procedure

     UNIT      - What disk procedure is on

(System prints out the OCL statements in the procedure.)

MODIFY         - RUN  ---------> Tells system the OCL statements are
                                 complete and correct.

INCLUDED STATEMENTS

(System prints out your instructions for the Disk Sort program.)

MODIFY         - RUN  ---------> Tells system the included statements
                                 are complete and correct.
```

REMEMBER...

- The MODIFY-RUN statement tells the system to run the job.
- The definition of running the job is different for each OCL cycle.
- The CALL cycle has two MODIFY-RUN statements when instructions for a system program have been included in the procedure you're calling.

After the operator presses one of the end-of-statement keys, the system checks the statement for errors. If the system finds an error, it prints an *error message*. The error message tells you what's wrong with the statement.

When there is a mistake in an OCL statement, the system prints an error message directly below the statement.

```
READY        - LOAD

LOAD NAME    - MYPROG

     UNIT    - G1 <---- Error (The only valid responses are F1, F2, R1, or R2.)

MESSAGE #03 - INVALID UNIT SPECIFIED <---- Error Message
```

Some of the messages are more complicated. To explain the more complicated messages and to give you and the operator suggestions for correcting OCL errors, the *IBM System/3 Model 6 Operation Control Language and Disk Utility Programs Reference Manual,* GC21-7516, lists all the OCL error messages, explains them, and suggests possible solutions.

After the system prints an error message, it either reprompts the keyword or, if the error is a very serious one, cancels the job. If the job is canceled, the message explains why. For example, one of the messages is: MESSAGE #39 – ERRORS IN PROCEDURE – JOB CANCELED.

Following a job canceled message, the system prompts READY, and the operator can either try to run the job again or start a new job.

If the system prompts a keyword the second time, and the operator again makes an invalid response, the system prompts the keyword a third time. The system will continue to reprompt the keyword until the operator makes a valid response.

```
READY          - BUILD

BUILD NAME     - MSALES

      UNIT     - F3 <------------- Invalid Response

MESSAGE #03    - INVALID UNIT SPECIFIED

      UNIT     - D2 <------------- Invalid Response

MESSAGE #03    - INVALID UNIT SPECIFIED

      UNIT     - F2 <------------- Valid Response

LOAD NAME      - <------------- Keyword prompting continues
```

REMEMBER. . .

- The system checks each OCL statement for errors.
- If a statement contains an error, the system prints an error message directly below the statement specifying what is wrong with the statement.
- After the system prints an error message, it either reprompts the keyword or cancels the job.

If you want to test yourself on the material presented in chapters 10 and 11, see the self-test questions in Part V of this manual.

You can use your own procedure or an IBM-supplied compile procedure to compile your RPG II source program (See *Chapter 6. The Compile Keywords* for a discussion of the compile keywords and OCL cycles used to compile your RPG II program.) There are two IBM-supplied compile procedures for compiling an RPG II program: RPG and RPGB.

For example, say you wish to compile a source program PAYROL (a payroll update) located on R2. When you use the IBM-supplied procedure RPGB to compile your RPG II program, the information needed to compile your program, except for the response to compile keywords SOURCE and UNIT, has previously been keyed in. The responses to these keywords are supplied by the operator each time the procedure is run. The OCL sequence would look like the following:

```
EADY                          - CALL (P/S)
00    CALL  NAME              - RPGB (P/S)
01          UNIT              - F1 (P/S)
*****************************************
10    LOAD  NAME              - $RPG
11          UNIT              - R2
20    COMPILE OBJECT          - R1
21          SOURCE            - PAYROL (P/S)
22          UNIT              - R2 (P/S)
30    FILE  NAME              - $WORK
31          UNIT              - R1
32          PACK              - 111111
33          TRACKS            - 20
40    FILE  NAME              - $SOURCE
41          UNIT              - R1
42          PACK              - 111111
43          TRACKS            - 20
*****************************************
ODIFY
UN (P/S)
```

Procedure displayed by CALL cycle.

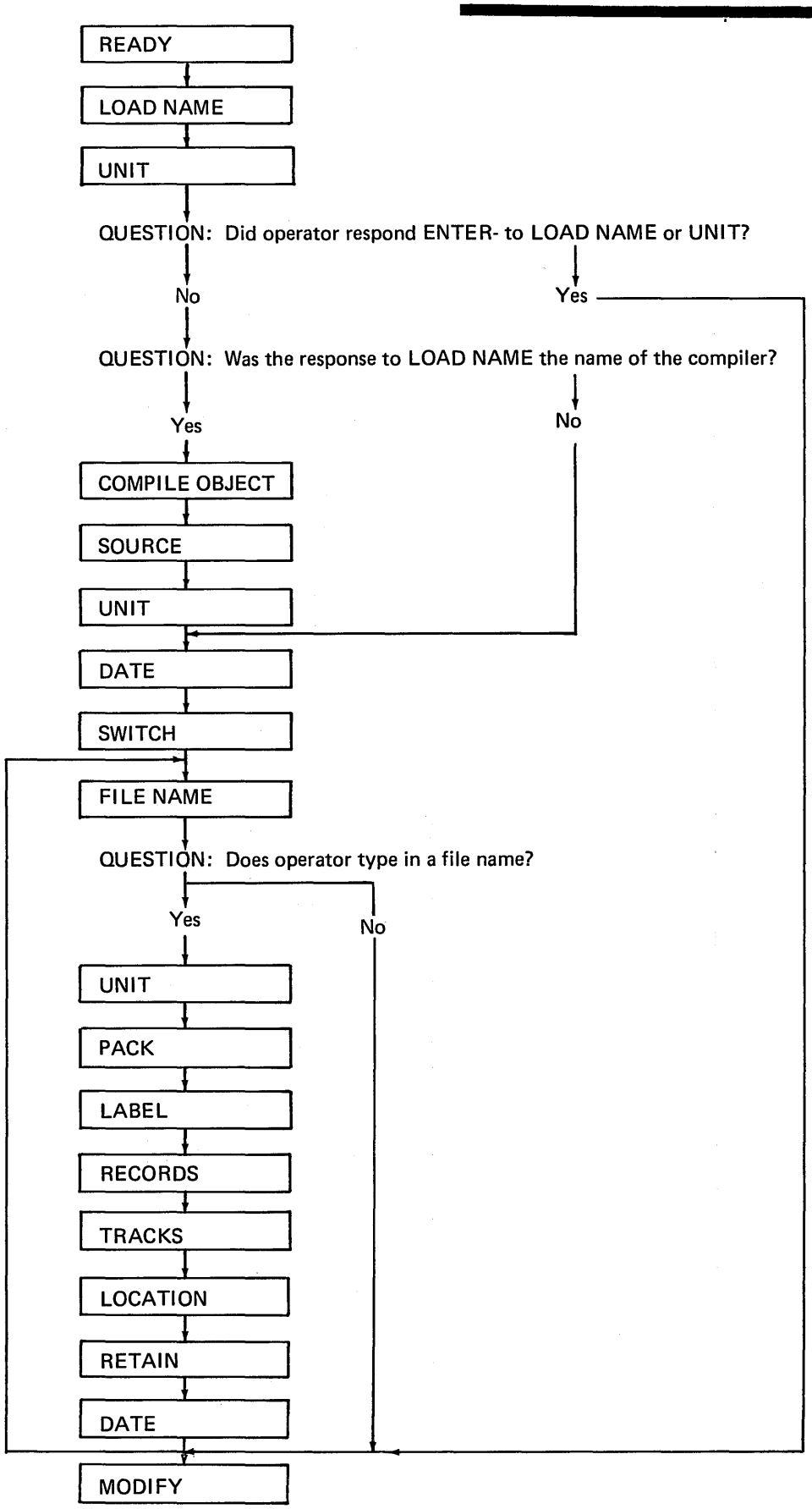
OCL Statements.

Shaded areas are operator responses. The PROG START (P/S) key is pressed after each response.

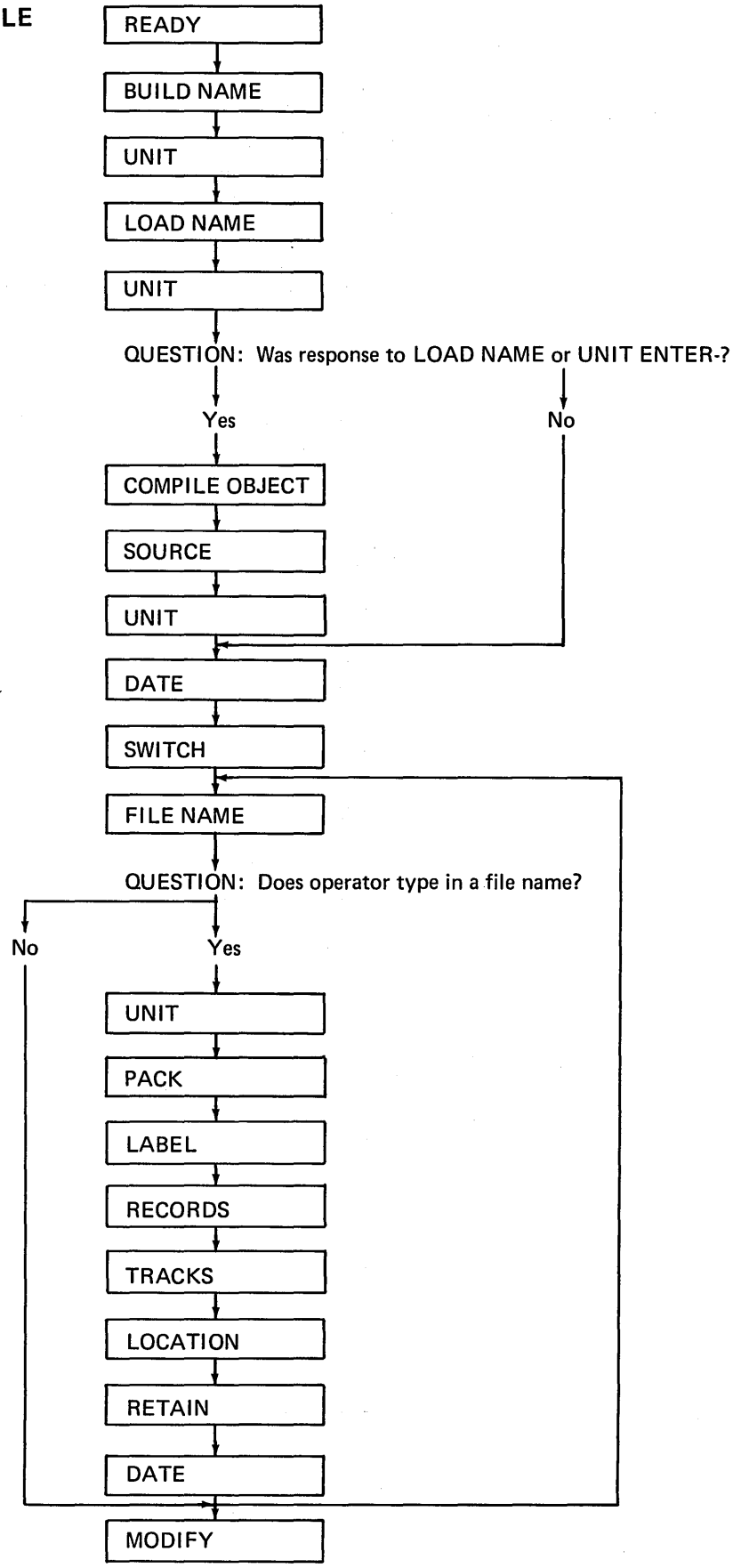
The CALL cycle is being used in this example to call the IBM-supplied compile procedure RPGB. Statements 010-043 are the procedure used to compile your RPG II program. LOAD NAME - $RPG tells the system you want to use the RPG II Compiler. You must respond to only SOURCE and UNIT for the compile keywords. The object program is to be placed on R1. The program to be compiled, PAYROL, is located on R2. The files required by the compiler ($WORK, $SOURCE) are defined for you. All other responses are included in the procedure. You may modify the procedure in the manner specified in *Chapter 9. MODIFY.*

If you had used the IBM-supplied procedure RPG, the compile keywords wouldn't have been in the procedure. The object program would automatically be placed on the same unit as the compiler. The source statements would come from the system input device. (This procedure is normally used only when an IBM 5496 Data Recorder is attached to the Model 6 as the system input device.)

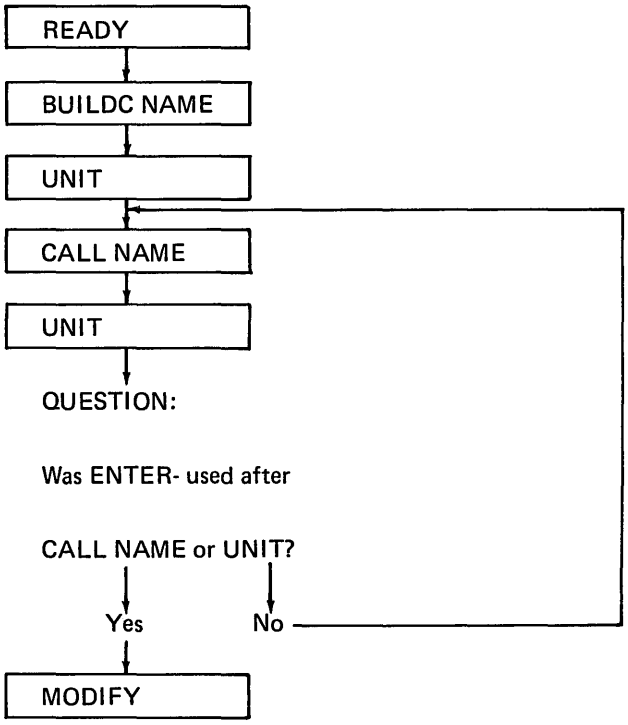
THE LOAD CYCLE



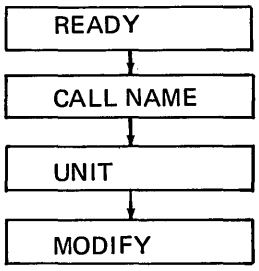
THE BUILD CYCLE



THE BUILDC CYCLE



THE CALL CYCLE



PART III. DISK UTILITY PROGRAMS

CHAPTER 14. INTRODUCTION TO DISK UTILITY PROGRAMS

Every method of data processing requires a certain amount of maintenance work to keep it in good running order. For example, you must make back-up copies of important files, and remove out-of-date files. The Disk Utility programs are a collection of maintenance programs to serve your data-processing system. The Disk Utility programs are:

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

You might use one of the preceding utility programs to:

- Prepare disks for use.
- Replace defective tracks.
- Replace incorrect data on a track.
- Print VTOC (volume table of contents) information.
- Delete files from a disk.
- Copy or print files.
- Maintain system libraries.

CHAPTER 15. DISK INITIALIZATION PROGRAM

Disks that are being used for the first time must be prepared for use. This process is called *initialization*. You can also use a disk that has been used before by reinitializing that disk (any data on the disk is destroyed). You use the Disk Initialization program to perform initialization.

FUNCTIONS

Initializing a disk involves:

- Naming the disk.
- Writing track and sector addresses on the disk.
- Checking for defective tracks.
- Assigning alternate tracks to any defective tracks.

Naming A Disk

You must name every disk you intend to use. The operator uses this name to ensure that the correct disks are being used for a job. He supplies the disk name in either OCL statements or program control statements. The system checks this name against the name stored as identification on the disk pack. If the names don't match, a halt occurs and a message is printed to the operator. The operator may then change disks. All this must happen before a Model 6 program can use a disk.

Writing Track and Sector Addresses

A disk contains 200 or 400 tracks, each of which is divided into 24 sectors. An area at the beginning of every track and sector is set aside for an address. These addresses are necessary for locating data.

Track and sector addresses are not written on disks when the disks are manufactured. You must do this before you use the disks. The Disk Initialization program does it for you.

Checking for Defective Tracks (Surface Analysis)

The Disk Initialization program checks the condition of tracks. It does this by writing data on the tracks, then reading and checking the data to ensure it was recorded properly. If the check shows that the data is incorrect, the track on which the data was written is considered defective. This process is called *surface analysis.*

Assigning Alternate Tracks

If a defective track is found during surface analysis, an alternate track is assigned to it. The sole purpose of the alternate track is to act as a substitute for the defective track. Model 6 programs attempting to use the defective track will automatically use the alternate instead.

Every disk has six alternate tracks. Therefore, a maximum of six defective tracks may be assigned alternates on a disk. If there are more, the disk is considered unusable.

OPTIONS

The Disk Initialization program allows you the following options:

- You may choose one of three types of initialization: primary, secondary, or clear.
- You may initialize up to three disks during the same program run.
- During primary initialization, you may decide whether to erase alternate track assignments already on the disk or leave them assigned.
- You may use up to ten characters, in addition to the disk name, to further identify a disk.
- You may specify the number of times you want the program to do surface analysis.

You specify the options you want in control statements (see *Control Statements* in this chapter).

Type of Initialization

The program offers three types of initialization: primary, secondary, and clear. The type you choose determines the portion of the disk that will be initialized. The portions of a disk that can be initialized depend on the data-storage capacity of your disk drive.

Disk drives of differing storage capacities are available for your system. 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.

If you increase the capacity of your disk drives, more tracks on your disks become available for use. These additional tracks must be initialized before being used. The three types of initialization allow you the following options according to type.

- Primary or clear—initializing all tracks corresponding to the new capacity, including any that were previously initialized.
- Secondary—initializing only the additional tracks made available by the increased capacity.

Primary Initialization

In primary initialization, all disk tracks corresponding to the specified drive capacity are initialized. Tracks previously initialized are reinitialized. Any data on the tracks is destroyed.

Primary initialization is required for new disks. You may also use it for disks that have been initialized before, provided they contain no libraries, temporary data files, or permanent data files. You must delete libraries using the allocate function of the Library Maintenance program and delete permanent and temporary data files using the File Delete program.

Secondary Initialization

Secondary initialization is used only for disks that were initialized on disk drives of lesser capacity then the ones you are now using. It's normally used for disks containing information in the previously initialized area, such as libraries, temporary files, and permanent files that you want to keep.

In secondary initialization, only the additional tracks made available by the increased capacity are initialized. The remainder of the disk isn't disturbed.

Clear Initialization

Clear initialization is used only on disks which can't be used because they have invalid pack labels or some other unrecoverable disk error. All tracks corresponding to the drive capacity are initialized, and previously initialized tracks are reinitialized. All libraries, temporary data files, and permanent data files are destroyed. Therefore, you should avoid using this type of initialization.

Number of Disks

The Disk Initialization program can initialize a maximum of three disks during one program run. The type of initialization you specify for a program run applies to all disks being initialized during that run. The disks, however, must be mounted at the same time. You can't, for example, initialize more than one removable disk on a given drive during the same program run.

Erasing Alternate Track Assignments

You can use primary or clear initialization to reinitialize disks that have been used. However, alternate track assignments could exist on such disks. The Disk Initialization program, therefore, gives you the option of:

- Erasing existing alternate track assignments and checking the condition of all tracks.
- Leaving existing alternate track assignments and checking only those tracks to which alternates are not assigned.

The option you choose applies to all disks being initialized during the program run.

Additional Disk Identification

When you name a disk during primary or clear initialization, you can use up to ten characters, in addition to the disk name, to further identify the disk. The additional identification is strictly for your use. It is not used by the checking programs to ensure that the right disks are being used.

If you use the File and Volume Label Display program to print VTOC (volume table of contents) information from a disk, the additional identification is printed with the disk name.

Surface Analysis Option

You can tell the Disk Initialization program to perform surface analysis from 1 to 255 times before judging whether or not tracks are defective. A track must successfully complete every check before being judged usable. If incorrect data is detected during surface analysis, the track on which the data was written is judged defective and an alternate is assigned to it.

The number of times you specify surface analysis to be performed applies to all disks being initialized during the program run. The time required for initialization is increased if you request surface analysis to be performed more than once.

CONTROL STATEMENTS

You must supply the following control statements to specify the program options you want:

1. *UIN statement*—indicates the type of initialization, the number of disks being initialized, the number of times you want surface analysis performed, and whether or not you want previous alternate track assignments erased. One UIN statement is required per program run.

2. *VOL statement*—indicates the name you assign to the disk, plus any additional identification you want to give the disk. The VOL statement applies to primary and clear initialization only. One is required for every disk you initialize.

3. *END statement*—indicates the end of control statements.

EXAMPLE

As an example, suppose you wanted to reinitialize two disks because you no longer needed the information stored on them. The following example shows how you would use the Disk Initialization program to do this.

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

OCL LOAD Sequence

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. (Circled areas are operator responses.)

```
MODIFY
RUN
   ENTER '//' CONTROL STATEMENT
// UIN UNIT-'F2,R2',TYPE-PRIMARY,ERASE-YES
   ENTER '//' CONTROL STATEMENT
// VOL PACK-INVOIC,ID-013077
   ENTER '//' CONTROL STATEMENT
// VOL PACK-3333
   ENTER '//' CONTROL STATEMENT
// END
```

Message printed by Disk Initialization program.

Control statement supplied by operator.

Sequence repeats until operator enters // END.

Explanation

The first group of statements (READY, 010-040, MODIFY, RUN) are the statements necessary to load the Disk Initialization program ($INIT). After the RUN response to the MODIFY keyword, the program prints a message requesting utility control statements. At this point in the example, you must specify the options you need to reinitialize two disks.

To reinitialize a disk, you must specify TYPE-PRIMARY and the disk unit. In this case, the disks you want to reinitialize are located on F2 and R2. For each disk specified in the UNIT parameter of the UIN statement, you must supply a VOL statement (if type is primary) and assign pack names to the disks. In this case, the pack names are INVOIC and 3333. You may also further identify a disk by assigning an ID (you assigned ID-013077 to disk INVOIC).

When you reinitialize the disks specified in this example, you are treating them as if they are new disks by specifying ERASE-YES. Therefore, you are allowed to erase all existing alternate track assignments and to check the condition of all tracks.

A // END statement will terminate prompting of the program request for control statements. The Disk Initialization program will then execute the job indicated by your control statements.

CHAPTER 16. ALTERNATE TRACK ASSIGNMENT PROGRAM

Sometimes a disk track causes a reading or writing error during a job and an *alternate track* must be assigned to replace the defective track. The process of assigning an alternate track is performed by the Alternate Track Assignment program.

FUNCTIONS

The process of assigning an alternate track consists of:

- Writing track addresses on disk.
- Checking for defective tracks.
- Printing all track sectors that contain incorrect data.
- Assigning an alternate track.

Writing Track Addresses

Any time a track causes reading or writing errors during a job, the system stops the program currently in operation and writes the track address in a special area on the disk. All disks contain such an area. The program can then locate a track by using the addresses stored in this area. As long as there are alternate tracks available for use, assignment can be done for all the tracks identified in this area.

Checking For Defective Tracks

The Alternate Track Assignment program uses a procedure called *surface analysis* to test the condition of tracks. It transfers test data from the suspected track to an alternate track and reads and checks the data to ensure that it was recorded properly. If the suspected track is defective, this is the alternate track that is assigned.

Printing Sectors Containing Incorrect Data

The alternate track assigned if a track is defective may contain incorrect data from the defective track. When the Alternate Track Assignment program is reading data from the defective track, it prints all track sectors that may contain data causing reading errors. To correct the errors on an alternate track, use the Alternate Track Rebuild program.

Assigning An Alternate Track

An alternate track is assigned if a track is defective. The alternate track is then automatically used any time the program attempts to use the defective track.

OPTIONS

The Alternate Track Assignment program gives you the following options:

- You may choose one of three types of assignment—conditional, unconditional, or cancel prior.
- You may use up to six alternate tracks on every disk.
- You may specify the number of times you want the program to do surface analysis.

You specify the options you want in control statements (see *Control Statements* in this chapter).

Type of Assignment

The program offers three types of assignment: conditional, unconditional, and cancel prior. The three types of assignment allow you the following options according to type.

- Conditional—testing the condition of a track and assigning an alternate if it is defective.
- Unconditional—assuming a track is defective and assigning an alternate.
- Cancel prior—canceling an alternate track assignment.

Conditional Assignment

Conditional assignment is the normal use of the Alternate Track Assignment program. When conditional assignment is specified, a track is tested for errors (surface analysis). If the track is defective an alternate is assigned. Prior to surface analysis, the program transfers the data from the suspected track to the alternate track that is used if the suspected track is found defective.

Unconditional Assignment

Unconditional assignment is used when the program has attempted to use conditional assignment, but the suspected track was not found to be defective even though it had caused occasional reading or writing errors. For this reason you should assign an alternate track using unconditional assignment. Alternate tracks are assigned without first testing the condition of the tracks suspected of being defective. (A conditional assignment is forced after an unconditional request to check any other tracks that previously caused errors.)

Cancel Prior Assignment

Cancel prior assignment is used to free an alternate track for use with another track if there are no other alternates available. Canceling an assignment involves transferring the data from an alternate track back to the track to which the alternate was assigned. Prior to transferring the data back to the original track, the Alternate Track Assignment program tests the condition of the original track. If the track is found defective, the program stops and one of three options is taken:

- You leave the assignment as it is but continue checking other assignments (if there are any), or the program ends.
- You cancel the assignment regardless of the condition of the original track.
- You test the track again.

You must run the File and Volume Label Display program to determine to what tracks alternates are assigned.

Number of Alternate Tracks

There are six tracks on every disk that can be used as alternates. These tracks, in addition to tracks 0 and 1, can't be replaced; that is, they can't have an alternate assigned to them.

Surface Analysis Option

You can tell the program to do surface analysis from 1 to 255 times before judging whether or not tracks are defective. A track is judged usable only after successfully completing every check. If at any time during surface analysis incorrect data is found, the track on which the data was written is judged defective, and an alternate is assigned to it.

CONTROL STATEMENTS

You must supply the following control statements to specify the program options you want:

1. *ALT statement*—indicates the name and unit of the disk containing the defective track, the number of times you want surface analysis done, and the tracks to which you want to assign alternates or for which you wish to cancel assignment of an alternate track.

2. *END statement*—indicates the end of control statements.

CHAPTER 17. ALTERNATE TRACK REBUILD PROGRAM

An alternate track may contain some incorrect data. In order to correct this data, you must use the Alternate Track Rebuild program.

FUNCTIONS

The process of correcting data consists of:

- Locating incorrect data.
- Replacing incorrect data.

Locating Incorrect Data

The Alternate Track Assignment program prints a listing of all track sectors that may contain incorrect data. You will find, on the listing, the name of the disk, the track and sector numbers of the area suspected of containing incorrect data, and the data from these sectors.

Replacing Incorrect Data

The Alternate Track Rebuild program will replace the number of characters you indicate in the positions you indicate. You must key the new characters in hexadecimal form. These characters are called *substitute data.*

OPTIONS

The Alternate Track Rebuild program gives you the following options:

- You may correct as many characters as you wish on one track.
- You may correct data on more than one track.

You specify the options you want in control statements (see *Control Statements* in this chapter).

Number of Characters

You may replace from 2 to 256 characters on one track in one run. You can do this by replacing all the characters (including correct data) or just groups of incorrect data.

Number of Tracks

The Alternate Track Assignment program prints the track and sector numbers for those areas that contain incorrect data. You can correct one or more of these tracks in one program run. The possible tracks you can correct are 8 through 405 and the sectors are 0 through 23. Tracks 0 through 7 can't be corrected.

CONTROL STATEMENTS

You must supply the following control statements to specify the program options you want:

1. *REBUILD statement*—indicates the name of the disk containing incorrect data, the track and sectors to be corrected, and the position and number of characters to be replaced. A REBUILD statement is needed for each group of characters to be corrected. The substitute data follows each REBUILD statement.

2. *END statement*—indicates the end of control statements.

CHAPTER 18. FILE AND VOLUME LABEL DISPLAY PROGRAM

You may need to obtain specific information about a file; find space available for libraries or new files; or check the contents of a disk for libraries, temporary data files, or permanent data files. In order to do any of these, you need information contained in the *volume table of contents* (VTOC). To obtain this information you must use the File and Volume Label Display program.

FUNCTIONS

This program allows you to:

- Print VTOC information.
- Print headings for file information.

Print VTOC Information

The VTOC is an area on disk that contains information about the contents of the disk. Every disk contains a VTOC. The File and Volume Label Display program allows you to access this information.

Print Headings

If the file information you requested from the VTOC overflows onto another page, the program prints the headings for the information at the top of the next page. It will do this for each succeeding new page.

OPTIONS

The File and Volume Label Display program gives you the following options:

- You may specify a printout of the entire VTOC or a printout of VTOC information for certain data files.
- You may specify up to 20 file names in one run.

You specify the options you want in control statements (see *Control Statements* in this chapter).

Entire Contents of VTOC

There are many reasons why you may want to print the entire VTOC. You may want to check which tracks are assigned alternates or how many alternate tracks are still available for use. You may also want to check the boundaries of libraries or check for permanent or temporary data files.

File Information Only

You may request information for specific files. You may want this information to find out file names, file designations, or disk areas reserved for files. You may also use it to determine the relationship of multivolume files.

Number of File Names

When you specify a file name, you must use the name that identifies the file in the VTOC. You are allowed to specify up to 20 file names in one program run.

CONTROL STATEMENTS

You must supply the following control statements to specify the program options you want:

1. *DISPLAY statement*—indicates whether you want the entire VTOC or specific file information from the VTOC. It also indicates the unit of the disk containing VTOC information.

2. *END statement*—indicates the end of control statements.

CHAPTER 19. FILE DELETE PROGRAM

You may find that you no longer need the information in a file. You can free the space in a file for use by new files by using the File Delete program.

FUNCTIONS

This program allows you to:

- Eliminate file references in the VTOC.
- Erase information in a file.

VTOC File References

The File Delete program allows you to remove the VTOC references to a file by removing the reference. However, the file reference is not physically removed from the VTOC until normal end of job has occurred.

Erase File Information

You may erase a file from the disk as well as removing the file reference in the VTOC. This involves erasing the information contained in the file. Its space is then made available for any new files.

OPTIONS

The File Delete program gives you the following options:

- You may choose to delete files in one of two ways: remove or scratch.
- You may delete some or all files from a disk.
- You may specify up to 52 file names in one job.

You specify the options you want in control statements (see *Control Statements* in this chapter).

Deleting a File

If you wish to delete a permanent file, you must use the File Delete program. If you delete a temporary file, you may use either the File Delete program or change the file designation when you use the file. You may either remove or scratch a file.

Removing a File

When you remove a file from a disk, you are removing the file reference from the VTOC. You may also erase the file from the disk, leaving its area available for use by other files.

Scratching a File

The File Delete program allows you to scratch a file if you find you may need to reference it later. When you scratch a file, the VTOC reference is not removed but changed to designate a scratch file. You can use the file until a permanent file is created in its place.

Number of Files

You may remove some or all files on a disk. If a file name applies to more than one file, all the files with that name are deleted. You can keep this from happening by identifying the files with both name and date.

Number of File Names

You may specify as many file names as the control statement will allow. If you specify more, you must use more than one statement. However, you are only allowed to specify 52 file names in one job.

CONTROL STATEMENTS

1. *REMOVE statement*—indicates the name and unit of the disk, what files are to be removed, and whether or not you are erasing the data for the file.

2. *SCRATCH statement*—indicates the name and unit of the disk and what files you wish to scratch.

3. *END statement*—indicates the end of control statements.

You may need to check records in a file for errors. In order to do this you need to *print a copy* of the file. It is important to provide a reserve disk or file for disks containing libraries or permanent data files in case something happens to the original disk or file. You can *copy* the disk or file using the Disk Copy/Dump program.

FUNCTIONS

Copying a disk or file involves:

- Identifying disk or file locations.
- Using a work area.

Printing a file involves:

- Identifying the portion to be printed.
- Printing record key or relative record numbers.

Disk or File Location

In order to copy a disk or file, you must specify the unit on which the disk or file is located and the unit to which it is to be copied. You can copy from one disk to another or from one area to another on the same disk (the latter applies to only part of a disk or file). Information for copying a file is contained in your responses to the OCL keywords prompted for the Disk Copy/Dump program (see the *IBM System/3 Model 6 Operation Control Language and Disk Utility Programs Reference Manual,* GC21-7516).

Using a Work Area

When you are copying a disk or file to another disk but have only one disk drive, you must use available space on the fixed disk on drive one. The disk you copy from must be a removable disk. The information from the disk you are copying is transferred to the available space on the fixed disk where it remains until another removable disk is mounted. This is the removable disk to which the information is copied.

If you are copying a file from one area on a removable disk to another area on the same disk you needn't use a work area on the fixed disk.

Printing a Portion of a File

You can print all or part of a file.

Record Keys and Relative Record Numbers

For indexed files the Disk Copy/Dump program will print each record key (used to access the record) followed by the contents of the record. The records are printed either in the order their keys appear in the index portion of the file or as they appear in the file itself. For sequential and direct files, a record is printed with its relative record number (used to access the record) preceding the record. The records are printed in the order they appear in the file.

OPTIONS

The Disk Copy/Dump program allows you the following options:

- You may copy an entire disk or a file.
- You may print part or all of a file.
- You may delete records from a file.
- You may reorganize a file.

You specify the options you want in control statements (see *Control Statements* in this chapter).

Copying and Printing

You may specify any of the following copy or print combinations:

- Copy an entire disk.
- Copy a data file.
- Copy and print a data file.
- Copy a data file, but print only part of the file.
- Print an entire data file.
- Print only a part of a data file.

Deleting Records

If you wish to delete records from a file while copying or printing, you must indicate the type of record you wish to omit. To do this, you must specify the *identifying character* (any of the standard System/3 character set except commas, apostrophes, and blanks) and the position of the character in the records (maximum position 999). The records that are deleted are printed. When the records of a file are being printed, the deleted records are indicated.

Reorganizing a File

When you are copying an indexed file you can reorganize it. The records in the data portion are put in the same order as their index keys leaving the original of the file you are copying unaffected. If you are both copying and printing an indexed file, you must specify reorganization.

CONTROL STATEMENTS

You must supply the following control statements to specify the program options you want:

1. *COPYPACK statement*—indicates that an entire disk is to be copied. It contains the unit of the disk to be copied and the disk to which the copying is being done.
2. *COPYFILE statement*—indicates that all or part of a data file is being copied or printed or both, whether the file is to be reorganized, and whether any records are to be deleted. It also allows you to specify if you want a work area.
3. *SELECT KEY statement*—indicates, according to record keys, which part of an indexed file you want printed.
4. *SELECT RECORD statement*—indicates, according to relative record numbers, which part of a file you want printed.
5. *END statement*—indicates the end of control statements.

Your programs are stored on disk in an area called a *library.* You can update or add new entries in this library. In order to do so, you must use the Library Maintenance program.

FUNCTIONS

The Library Maintenance program has four functions:

- Allocate
- Copy
- Delete
- Rename

Allocate

The allocate function of the Library Maintenance program allows you to:

- Create libraries.
- Change the size of libraries.
- Delete libraries.
- Reorganize libraries.

Creating Libraries

Creating a library involves:

- Assigning a library to a disk.
- Assigning space for the library directory.
- Using a work area.

Assigning a Library to a Disk. You are allowed one source and one object library per disk. The libraries can be located anywhere on the disk where space is made available as long as the source library precedes the object library. You needn't have both libraries for a disk.

Assigning Space for the Library Directory. The Library Maintenance program creates a separate directory for each library. A *directory for a source or object library* contains information concerning each library entry. This information includes the name and location of the library entry. For a source library, the first two sectors of the first track are assigned to the directory. For an object library which includes system programs, the first three tracks are assigned to the directory. If system programs are not included, only the first track is assigned to the directory.

Another type of directory, the system directory, is also created by this program. The *system directory* contains information concerning the libraries and their directories. This information includes the size of and available space in the libraries and their directories. The system directory is contained in the volume label on any disk pack.

Using a Work Area. You must use a work area when you want to create a source library, but there is not enough available space preceding the object library. The object library entries are temporarily stored in the work area while that library is being moved to make space available for the source library.

Changing the Size of Libraries

Changing the size of a library involves:

- Moving the object library when increasing the source library.
- Moving the object library when decreasing the source library.
- Moving the object library when increasing or decreasing the object library.
- Reorganizing the libraries.

Moving the Object Library when Increasing the Source Library. The object library immediately follows the end position of the source library. If an object library is present, therefore, the source library cannot be increased in size without moving the object library. The object library must be moved to a work area temporarily while the source library is increased. When moved back from the work area, the object library immediately follows the new end position of the source library.

Moving the Object Library when Decreasing the Source Library. When the source library is decreased in size, its end location moves. If there is an object library it is moved so that it immediately follows the source library. The space that results from the decrease in size of the source library is shifted to follow the object library.

Moving the Object Library when Increasing or Decreasing the Object Library. The end location of the object library is moved when the object library increases or decreases in size. If the object library decreases, additional space is made available for file usage following the object library.

Reorganizing the Libraries. Any time you change the size of one of your libraries, the program reorganizes the library. See *Reorganizing Libraries* in this chapter.

Deleting Libraries

When you delete a library, you are making the area occupied by the library available for other use.

You are restricted in deleting object libraries. You can't delete object libraries containing system programs that control program loading. Also, you can't delete the object library from which the Library Maintenance program was loaded.

Note: It is important to remember that when you delete a source library, the object library is not moved. When you create a source library after deleting it, the same space may not be used for the source library. The object library (if there is one) must be moved to make space available for the source library.

Reorganizing Libraries

Reorganizing a library involves:

- Relocating source library entries.
- Relocating object library entries.
- Using a work area.

Relocating Source Library Entries. When you delete a source library entry, the area is used for new entries. Some times not all of an entry will fit into this vacated area. It must be continued in the next available area. Reading separated entries takes more time than if the entire entry were located in one area. Therefore, the Library Maintenance program relocates each entry so that it occupies one area only.

Relocating Object Library Entries. When you add an entry to the object library in an area where an entry has been deleted, there may be space leftover in this area. The Library Maintenance program relocates entries so that these gaps are eliminated. By doing this, you save space that may be used for other new entries.

Using a Work Area. The Library Maintenance program must use a work area when reorganizing the libraries. Library entries are temporarily stored in the work area during reorganization.

Copy

The copy function of the Library Maintenance program allows you to copy:

- From reader-to-disk (add or replace entries)
- From disk-to-disk (copy entries)
- From disk-to-printer (print entries)
- From disk-to-card (punch entries)
- From disk-to-printer-and-card (print and punch entries)

Copying a library entry involves:

- Identifying the location of an entry.
- Identifying an entry.
- Removing and reinserting blanks and duplicate characters.
- Compressing object programs and routines.

Identifying the Location of an Entry. An entry may be read from either the system input device (keyboard or card reader) or from disk. It can be copied to disk, printer, or cards.

Identifying an Entry. Entries are identified by their type and name. Entries that can be copied include source library, object library, and system directory entries. A name identifies specific entries within the library or directory. You can also further identify an entry by designating whether it is temporary or permanent. This allows the program to make a check before replacing an entry.

Removing and Reinserting Blanks and Duplicate Characters. Source statements and procedures are placed in the source library. Before source statements are put in the source library, blanks and duplicate characters are removed to save space. When the source statements are copied, blanks and duplicate characters are reinserted. Procedures are left unchanged when placed in the source library.

Compressing Object Programs and Routines. Object programs and routines are compressed, then placed in the object library. When they are read from the object library, they remain compressed.

Delete

Deleting an entry involves:

- Identifying the location of an entry.
- Identifying an entry.

Identifying the Location of an Entry. The entry to be deleted is contained on disk.

Identifying an Entry. Entries are identified by their type and name. Entries that can be deleted include source library and object library entries. A name identifies the particular entry being deleted. You can also further identify an entry by designating whether it is temporary or permanent.

Rename

The rename function of the Library Maintenance program allows you to change the name of a library entry. Renaming an entry involves identifying the disk location of the entry to be renamed.

OPTIONS

The Library Maintenance program gives you the following options. You can:

- Assign as many tracks to the source library or object library as are available on the disk.
- Include system programs in the object library.
- Specify any of a number of types of entries to be copied, deleted, or renamed.
- Specify up to six characters for an entry name.
- Specify specific entries by name for the copy and delete uses of this program.

You specify the options you want in control statements (see *Control Statements* in this chapter).

Library Size

The maximum library size is the number of tracks in the available disk area. The minimum size for the source library is one track. The minimum size of the object library depends on whether it is to contain a minimum system. (A *minimum system* is made up of those system programs necessary to load and run programs.) If the object library contains a minimum system, the library can be no smaller than 30 tracks. Otherwise, the minimum is three tracks.

System Programs

You can include system programs in the object library. If you do, three tracks must be assigned to the library directory, and space must be assigned for a scheduler work area. (The *scheduler work area* is a work area for one of the system programs, the Scheduler.) The scheduler work area immediately precedes the object library. The Library Maintenance program automatically assigns this space.

Types of Entries

For the copy function of the Library Maintenance program, you can specify source statements, procedures, object programs, routines, and system directory entries.

You can delete or rename source statements, procedures, object programs, and routines.

Length of Name

The name you specify for an entry must not exceed six characters. It can be any of the System/3 characters except blanks and periods. The first

character must be alphabetic (A-Z, @, #, or $). It is recommended that you don't use a dollar sign, however, since many IBM programs begin with $. The same rules apply to constructing a new name for an entry.

Names of Entries

You can specify an entry or group of entries for the copy or delete uses of this program.

COPY FUNCTION OF LIBRARY MAINTENANCE

You Can Copy

- One library entry.
- Library entries with the same name.
- Library entries that begin with certain characters.
- All library entries.
- The minimum system.
- An IBM program.

You Can Print

- One library entry.
- Temporary or permanent library entries with the same name.
- Temporary or permanent library entries with names beginning with certain characters.
- All temporary or permanent library entries of a certain type.
- Directory entries for library entries of a certain type.
- All library and system directory entries.
- Only the system directory.

You Can Punch

- One library entry.
- Temporary or permanent library entries with the same name.
- Temporary or permanent library entries with names beginning with certain characters.
- Temporary or permanent library entries of a certain type.

You Can Print and Punch

- One library entry.
- Temporary or permanent library entries with the same name.
- Temporary or permanent library entries with names beginning with certain characters.
- Temporary or permanent library entries of a certain type.

DELETE FUNCTION OF LIBRARY MAINTENANCE

You Can Delete

- One library entry.
- Temporary or permanent library entries with the same name.
- Temporary or permanent library entries with names beginning with certain characters.
- Temporary or permanent library entries of a certain type.

CONTROL STATEMENTS

You must supply the following control statements to specify the program options you want:

1. *ALLOCATE statement*—indicates whether you wish to create, change the size of, delete, or reorganize the source or object library. It also indicates the unit of the disk you are using, how many tracks you want to assign to the object library directory, and where the work area is (if you use one). You can't use more than four ALLOCATE statements per job.

2. *COPY statement*—indicates whether you wish to add, replace, copy, print, or punch entries. It also indicates what entries are to be used, on what unit the entries are located, their destination, their designation, and a new name if you wish the copy to have a different name.

3. *DELETE statement*—indicates what entries are to be deleted, the unit they are on, and their designation.

4. *RENAME statement*—indicates what entries are to be renamed, the unit they are on, and the new name you wish to give to them.

5. *END statement*—indicates the end of control statements.

PART IV. SAMPLE JOBS

CHAPTER 22. SAMPLE JOBS

Part IV contains six sample jobs similar to what you might run. Each sample job is organized into three sections:

1. An introductory summary explaining the job.

2. OCL statements (and utility control statements where applicable) for the job.

3. Explanatory notes on individual statements in the job.

SAMPLE JOB 1. INITIALIZE DISK

You're going to use the Disk Initialization program (located on the fixed disk on drive one) to initialize the removable disk on drive one. You 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                NAME-                   $INIT (P/S)
011                     UNIT-                   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:

     - Initialize the entire pack.

     - 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

You'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. You want to put the compiled program in the object library on R1. Here are the OCL statements for the job.

```
READY-                                  CALL (P/S)
000 CALL            NAME-               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
- 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"

You'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 — (ENTER-)
  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.

- MODIFY

SAMPLE JOB 4. COPY FILE DISK TO DISK

You'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, you:

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 you 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)
Q51                   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 Disk Copy/Dump program is on F1.

- 020 DATE — (12/08/70)
  You'll use the current system date for the job.

- 030 SWITCH — (00000000)
  This program doesn't use external indicators, so the 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 the 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 you are creating a new file, you must respond to one of the space keywords (TRACKS and RECORDS). You specify 15 tracks because that's what you 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. MULTIFILE 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. You 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 F1.

```
READY-                                  BUILD (P/S)
000 BUILD            NAME-              WTR (P/S)
001                  UNIT-              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)
096                 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 R2.

- 020 DATE — (P/S)
  The date statement isn't part of the procedure.

- 030 SWITCH — (00000000) — (P/S)
  The external indicators aren't 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 R1.

- 094 RECORDS — 500 (P/S)
  Your output file contains up to 500 records.

- 096 RETAIN — P (ENTER-)
  You want to make this a permanent file. The ENTER- key caused DATE to be skipped and FILE NAME prompted.

- 100 FILE NAME — (P/S)
  You are finished with file statements, prompt MODIFY.

- RUN — Put the procedure in the source library.

SAMPLE JOB 6. MULTIFILE CALL

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

```
READY-                                  CALL (P/S)
000 CALL              NAME-             WTR (P/S)
001                   UNIT-             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
  You must supply the date for each day's input file because you gave a delayed response (?) at BUILD time. Thursday's date is entered even though you will delete the file later. A date should be entered to continue the cycle.
- MODIFY 050 — You delete the entire file for Thursday and enter a comment to explain why.
- RUN — Start the job.

PART V. REVIEW QUESTIONS

Part V includes review questions for chapters 2 through 11 of this manual. You should use these questions to check your understanding of the important concepts of these chapters.

The questions are divided into chapters; sometimes two chapters are included in one group of questions. The answers are organized in the same manner and follow the entire group of questions.

Questions

Chapters 2 and 3

1. _____ are required to supply information for the system to run jobs.
2. The system requests information by printing a _____ and the operator enters a _____.
3. What is the purpose of the end-of-statement keys?

Chapter 4

1. Why is the LOAD cycle called independent?
2. LOAD cycle OCL statements are saved. True or false?
3. A set of OCL statements written on disk is referred to as a _____ .
4. The purpose of the BUILD cycle is to _____ .
5. After the BUILD cycle is complete, the operator can run the _____ cycle to run the job.
6. The BUILDC cycle is used to build a _____ .
7. What is the advantage of the BUILDC cycle?

Chapters 5 and 6

1. What is the first keyword in every cycle?
2. What are the two functions of this keyword?
3. What response is made to prompt LOAD NAME?
4. What response is made to prompt BUILD NAME?
5. What is the meaning of the response to the prompt UNIT after
   a) LOAD NAME, and after
   b) BUILD NAME?
6. CALL NAME asks for the name of the _____ you want to use.

Chapters 7 and 8

1. What keyword calls for the disk unit on which an object program is to be written after compilation?
2. What are the three file keywords for which a response must be given prior to running a job?
3. You would respond to the prompt_____ if you want the system to calculate how much space is required for your file.
4. Why would you not respond to the prompt LOCATION?
5. In which cycle is a delayed response valid?
6. MODIFY is prompted when the operator presses PROG START after the _____ prompt.

Chapter 9

1. What are the two methods used to cancel a job?
2. How does the system indicate which statements may be corrected or deleted?
3. Additional operator instructions are provided by_____statements.
4. The LOAD cycle allows you to include instructions for a system program. True or false?

Chapters 10 and 11

1. What response to MODIFY tells the system to run the job?
2. Some CALL cycles will have two MODIFY statements. True or false?
3. What indication does the operator have that there is an error in an OCL statement?
4. How many times will the system reprompt a keyword following an error?

Answers

Chapters 2 and 3

1. OCL statements
2. keyword; response
3. The end-of-statement keys are pressed by the operator to indicate the end of his response.

Chapter 4

1. The LOAD cycle is called independent because you can run a job by responding just to the keywords in that cycle.
2. false
3. procedure
4. save the OCL statements for a job by writing them on disk.
5. CALL
6. chained procedure
7. It allows you to run a group of jobs without stopping between each one to supply OCL statements.

Chapters 5 and 6

1. READY
2. a) Tell the system which OCL cycle you want to use.
   b) Assign the logging device.
3. The name of the program you want to load into the processing unit.
4. The name of the procedure you are building.
5. a) The disk unit on which the program you want to run is stored.
   b) The disk unit on which the procedure you are building is to be stored.
6. procedure

Chapters 7 and 8

1. COMPILE OBJECT
2. FILE NAME, UNIT, and PACK
3. RECORDS
4. You wouldn't respond if you don't wish to determine where your file should be stored on disk. The system will then determine where the file is to go.
5. BUILD cycle
6. FILE NAME

Chapter 9

1. a) You can respond with CANCEL to the MODIFY prompt.
   b) You can respond with /* after any prompt.
2. The system outlines them with a border of asterisks.
3. comment
4. false

Chapters 10 and 11

1. RUN
2. true
3. The system prints an error message directly below the statement.
4. Until the operator gives a valid response.

After you decide which OCL cycle you want to use for your job and how you want to respond to all the keywords in the cycle, you must give this information to your operator. One way you can do this is by filling out an *Operator's OCL Guide.* (Copies of the *Operator's OCL Guide*, GX21-9126, are available from your local IBM branch office.)

The *Operator's OCL Guide* has three sections: the left-hand section for the printed keywords, the middle section for you to fill in the responses to those keywords, and the right-hand section to remind you of some of the programming considerations that apply to the keywords.

When you're filling out the guide, use the upper left-hand corner to identify the job. If your installation has more than one programmer, be sure to fill in the space for job programmer so that your operator will know who to call in case he has a problem with the job.

IBM

International Business Machines Corporation

System/3 Model 6

GX21-9126-0
Printed in U.S.A.

Job _____

Date ___ Job Identification ___

Programmer _____

OPERATION CONTROL LANGUAGE (OCL) GUIDE

Keywords	Responses		Considerations
READY	BUILD	OR LOAD	
000 BUILD NAME			Procedure Name
001 UNIT			F1, R1, F2 or R2
010 LOAD NAME			Columns 75-80 of RPG Control Card or System Program Name
011 UNIT			F1, R1, F2 or R2
020 DATE			mmddyy or ddmmyy
030 SWITCH			1-On, 0-Off, X-No Change
040 FILE NAME			Columns 7-14 of RPG File Description Specifications or Predefined Filename
041 UNIT			F1, R1, F2 or R2
042 PACK			Disk Name (Assigned by Disk Initialization Program)
043 LABEL			VTOC File Name (if different than response to FILE NAME)
044 RECORDS			1-999999 (Maximum Number of Records in File)
045 TRACKS			1-398 (Maximum Number of Tracks for this File)
046 LOCATION			8-405 Location of First Track of File
047 RETAIN			S-Scratch, T-Temporary, P-Permanent
048 DATE			mmddyy or ddmmyy
050 FILE NAME			Columns 7-14 of RPG File Description Specifications or Predefined File Name
051 UNIT			F1, R1, F2 or R2
052 PACK			Disk Name (Assigned by Disk Initialization Program)
053 LABEL			VTOC File Name (if different than response to FILE NAME)
054 RECORDS			1-999999 (Maximum Number of Records in File)
055 TRACKS			1-398 (Maximum Number of Tracks for this File)
056 LOCATION			8-405 Location of First Track of File
057 RETAIN			S-Scratch, T-Temporary, P-Permanent
058 DATE			mmddyy or ddmmyy
MODIFY			MODIFY OPTIONS 1. Enter RUN 2. Enter CANCEL 3. Correct Statement Enter Statement number Retype or delete (,) response 4. Create new Statement INCLUDE, LOG, FORMS, *(For Comments)
Keywords	Response		Considerations

Other Possible Entry
(Lines 020-058)
? for Delayed Response

What part of the OCL Guide you fill out depends on what cycle you're using for the job. For LOAD cycles, you use the right-hand side of the Response section. For BUILD cycles, you use the left-hand side of the Response section. For the two short cycles, BUILDC and CALL, you use the blank area at the bottom of the guide. (For the BUILDC and CALL cycles you must fill in *both* the keywords and the responses.)

IBM

International Business Machines Corporation
System/3 Model 6

GX21-9126-0
Printed in U.S.A.

Job _____

Date _____

OPERATION CONTROL LANGUAGE (OCL) GUIDE

Programmer _____

Keywords	Responses		Considerations
READY	BUILD	OR LOAD	
000 BUILD NAME			Procedure Name
001 UNIT			F1, R1, F2 or R2
010 LOAD NAME			Columns 75-80 of RPG Control Card or System Program Name
011 UNIT			F1, R1, F2 or R2
020 DATE			mmddyy or ddmmyy
030 SWITCH			1-On, 0-Off, X-No Change
040 FILE NAME			Columns 7-14 of RPG File Description Specifications or Predefined Filename
041 UNIT			F1, R1, F2 or R2
042 PACK	For BUILD cycle	For LOAD cycle	Disk Name (Assigned by Disk Initialization Program)
043 LABEL			VTOC File Name (if different than response to FILE NAME)
044 RECORDS			1-999999 (Maximum Number of Records in File)
045 TRACKS			1-398 (Maximum Number of Tracks for this File)
046 LOCATION			8-405 Location of First Track of File
047 RETAIN			S-Scratch, T-Temporary, P-Permanent
048 DATE			mmddyy or ddmmyy
050 FILE NAME			Columns 7-14 of RPG File Description Specifications or Predefined File Name
051 UNIT			F1, R1, F2 or R2
052 PACK			Disk Name (Assigned by Disk Initialization Program)
053 LABEL			VTOC File Name (if different than response to FILE NAME)
054 RECORDS			1-999999 (Maximum Number of Records in File)
055 TRACKS			1-398 (Maximum Number of Tracks for this File)
056 LOCATION			8-405 Location of First Track of File
057 RETAIN			S-Scratch, T-Temporary, P-Permanent
058 DATE			mmddyy or ddmmyy
MODIFY			MODIFY OPTIONS 1. Enter RUN 2. Enter CANCEL 3. Correct Statement Enter Statement number Retype or delete (,) response 4. Create new Statement INCLUDE, LOG, FORMS, *(For Comments)
For BUILD C and CALL cycles			

Other Possible Entry (Lines 020-058)
? for Delayed Response

FILLING OUT THE OCL GUIDE FOR LOAD CYCLE

If you're using a LOAD cycle for your job, cross out the BUILD response in the first line and enter your responses in the LOAD column. Here is how you would fill out the sheet if you were using the LOAD cycle to run a weekly inventory program (INVENT), which is located on the fixed disk on drive one (F1). The job uses one file (MASTER) which is written on the disk named VOL03. VOL03 is the removable disk on drive one (R1).

IBM

GX21-9126-0
Printed in U.S.A.

International Business Machines Corporation

System/3 Model 6

Job INVENTORY

Date 05/23/71

Programmer S. Smith

OPERATION CONTROL LANGUAGE (OCL) GUIDE

Keywords	Responses			Considerations
READY	~~BUILD~~	OR	LOAD	
000 BUILD NAME				Procedure Name
001 UNIT				F1, R1, F2 or R2 — Other Possible Entry
010 LOAD NAME			INVENT	
011 UNIT			F1	
020 DATE				
030 SWITCH				
040 FILE NAME			MASTER	
041 UNIT			R1	F1, R1, F2 or R2
042 PACK			VOL03	Disk Name (Assigned by Disk Initialization Program)
043 LABEL				VTOC File Name (if different than response to FILE NAME)
044 RECORDS				1-999999 (Maximum Number of Records in File)
045 TRACKS				1-398 (Maximum Number of Tracks for this File)
046 LOCATION				8-405 Location of First Track of File
047 RETAIN				S-Scratch, T-Temporary, P-Permanent
048 DATE				
050 FILE NAME			P/S	
051 UNIT				
052 PACK				
053 LABEL				
054 RECORDS				
055 TRACKS				
056 LOCATION				
057 RETAIN				S-Scratch, T-Temporary, P-Permanent
058 DATE				mmddyy or ddmmyy
MODIFY			RUN	

A blank line indicates that the operator's only response is to press the PROG START key.

The inventory program uses one file. When the system prompts FILE NAME the second time, pressing the PROG START key tells it to bypass the series of file keywords and prompt MODIFY.

MODIFY OPTIONS

1. Enter RUN
2. Enter CANCEL
3. Correct Statement
   Enter Statement number
   Retype or delete (,) response
4. Create new Statement
   INCLUDE, LOG, FORMS, *(For Comments)

FILLING OUT THE OCL GUIDE FOR BUILD CYCLE

When you use a BUILD cycle, cross out the LOAD response in the first line and enter your responses in the BUILD column. Here's how you would fill out the sheet to build a procedure named WKBILL (weekly billing) to produce a weekly billing report. Assume you want to put the procedure on R1. To produce the report you run the program BILLING (billing) which is on F1. The program uses two files: CUSTFILE (customer file) and ORDFLE (order file). Both files are on R1. The name of the disk is VOL05.

IBM

GX21-9126-0
Printed in U.S.A.

International Business Machines Corporation
System/3 Model 6

Job Weekly Billing Procedure

Date Ø6/Ø1/71

Programmer S. Smith

OPERATION CONTROL LANGUAGE (OCL) GUIDE

Keywords	Responses		Considerations
READY	BUILD	~~ORLOAD~~	
000 BUILD NAME	WKBILL		Procedure Name
001 UNIT	R1		F1, R1, F2 or R2
010 LOAD NAME	BILLNG		Columns 75-80 of RPG Control Card or System Program Name
011 UNIT	F1		F1, R1, F2 or R2
020 DATE			mmddyy or ddmmyy
030 SWITCH			1-On, 0-Off, X-No Change
040 FILE NAME	CUSTFLE		Columns 7-14 of RPG File Description Specifications or Predefined Filename
041 UNIT	R1		F1, R1, F2 or R2
042 PACK	VOLØ5		Disk Name (Assigned by Disk Initialization Program)
043 LABEL			VTOC File Name (if different than response to FILE NAME)
044 RECORDS			1-999999 (Maximum Number of Records in File)
045 TRACKS			1-398 (Maximum Number of Tracks for this File)
046 LOCATION			8-405 Location of First Track of File
047 RETAIN			S-Scratch, T-Temporary, P-Permanent
048 DATE			mmddyy or ddmmyy
050 FILE NAME	ORDFLE		Columns 7-14 of RPG File Description Specifications or Predefined File Name
051 UNIT	R2		F1, R1, F2 or R2
052 PACK	VOLØ5		Disk Name (Assigned by Disk Initialization Program)
053 LABEL			VTOC File Name (if different than response to FILE NAME)
054 RECORDS			1-999999 (Maximum Number of Records in File)
055 TRACKS			1-398 (Maximum Number of Tracks for this File)
056 LOCATION			8-405 Location of First Track of File
057 RETAIN			S-Scratch, T-Temporary, P-Permanent
058 DATE			mmddyy or ddmmyy
~~MODIFY~~			~~MODIFY OPTIONS~~
Ø6Ø FILE NAME	P/S		
MODIFY	RUN		

Other Possible Entry (Lines 020-058) ? for Delayed Response

The sheet has preprinted keywords for two files per cycle. You write in the third FILE NAME prompt.

4. Create new Statement
INCLUDE, LOG, FORMS, *(For Comments)

FILLING OUT THE OCL GUIDE FOR BUILDC OR CALL CYCLE

You use the bottom of the guide for either a CALL or BUILDC cycle. For these cycles you must fill in both the keywords and the responses. Here's how you would fill out the sheet to CALL the procedure WKBILL to run your weekly billing report.

IBM

GX21-9126-0
Printed in U.S.A.

International Business Machines Corporation
System/3 Model 6

Job *Weekly Billing*

Date *08/03/71*

Programmer *S. Smith*

OPERATION CONTROL LANGUAGE (OCL) GUIDE

Keywords	Responses			Considerations
READY	BUILD	OR	LOAD	
000 BUILD NAME				Procedure Name
001 UNIT				F1, R1, F2 or R2
010 LOAD NAME				Columns 75-80 of RPG Control Card or System Program Name
011 UNIT				F1, R1, F2 or R2
020 DATE				mmddyy or ddmmyy
030 SWITCH				1-On, 0-Off, X-No Change
040 FILE NAME				Columns 7-14 of RPG File Description Specifications or Predefined Filename
041 UNIT				F1, R1, F2 or R2
042 PACK				Disk Name (Assigned by Disk Initialization Program)
043 LABEL				VTOC File Name (if different than response to FILE NAME)
044 RECORDS				1-999999 (Maximum Number of Records in File)
045 TRACKS				1-398 (Maximum Number of Tracks for this File)
046 LOCATION				8-405 Location of First Track of File
047 RETAIN				S-Scratch, T-Temporary, P-Permanent
048 DATE				mmddyy or ddmmyy
050 FILE NAME				Columns 7-14 of RPG File Description Specifications or Predefined File Name
051 UNIT				F1, R1, F2 or R2
052 PACK				Disk Name (Assigned by Disk Initialization Program)
053 LABEL				VTOC File Name (if different than response to FILE NAME)
054 RECORDS				1-999999 (Maximum Number of Records in File)
055 TRACKS				1-398 (Maximum Number of Tracks for this File)
056 LOCATION				8-405 Location of First Track of File
057 RETAIN				S-Scratch, T-Temporary, P-Permanent
058 DATE				mmddyy or ddmmyy
MODIFY				MODIFY OPTIONS
READY	*CALL*			1. Enter RUN
CALL NAME	*WKBILL*			2. Enter CANCEL
UNIT	*R1*			3. Correct Statement Enter Statement number Retype or delete (,) response
MODIFY	*RUN*			4. Create new Statement INCLUDE, LOG, FORMS, *(For Comments)

Other Possible Entry (Lines 020-058) ? for Delayed Response

FILLING OUT THE OCL GUIDE FOR MORE THAN TWO FILES

Several of your jobs may use more than two files. By using both sides of the Response section, you can indicate responses for the keywords for four files. (This applies only when you're using either a LOAD or BUILD OCL cycle.)

IBM

International Business Machines Corporation

System/3 Model 6

GX21-9126-0
Printed in U.S.A.

Job _____

Date _____ **OPERATION CONTROL LANGUAGE (OCL) GUIDE**

Programmer _____

Keywords	Responses		Considerations
READY	BUILD	OR LOAD	
000 BUILD NAME			Procedure Name
001 UNIT			F1, R1, F2 or R2
010 LOAD NAME			Columns 75-80 of RPG Control Card or System Program Name
011 UNIT			F1, R1, F2 or R2
020 DATE			mmddyy or ddmmyy
030 SWITCH			1-On, 0-Off, X-No Change
040 FILE NAME			Columns 7-14 of RPG File Description Specifications or Predefined Filename
041 UNIT			F1, R1, F2 or R2
042 PACK			Disk Name (Assigned by Disk Initialization Program)
043 LABEL			VTOC File Name (if different than response to FILE NAME)
044 RECORDS	FILE	FILE	1-999999 (Maximum Number of Records in File)
045 TRACKS			1-398 (Maximum Number of Tracks for this File)
046 LOCATION			8-405 Location of First Track of File
047 RETAIN			S-Scratch, T-Temporary, P-Permanent
048 DATE			mmddyy or ddmmyy
050 FILE NAME			Columns 7-14 of RPG File Description Specifications or Predefined File Name
051 UNIT			F1, R1, F2 or R2
052 PACK			Disk Name (Assigned by Disk Initialization Program)
053 LABEL			VTOC File Name (if different than response to FILE NAME)
054 RECORDS	FILE	FILE	1-999999 (Maximum Number of Records in File)
055 TRACKS			1-398 (Maximum Number of Tracks for this File)
056 LOCATION			8-405 Location of First Track of File
057 RETAIN			S-Scratch, T-Temporary, P-Permanent
058 DATE			mmddyy or ddmmyy
MODIFY			MODIFY OPTIONS 1. Enter RUN 2. Enter CANCEL 3. Correct Statement Enter Statement number Retype or delete (,) response 4. Create new Statement INCLUDE, LOG, FORMS, *(For Comments)

Other Possible Entry (Lines 020-058) ? for Delayed Response

Here's how you would fill out the sheet to build a procedure with three files.

IBM

International Business Machines Corporation
System/3 Model 6

GX21-9126-0
Printed in U.S.A.

Job *Procedure for Monthly Report*

Date *12/07/71*

Programmer *S. Smith*

OPERATION CONTROL LANGUAGE (OCL) GUIDE

Keywords	Responses			Considerations
READY	BUILD	OR	LOAD	
000 BUILD NAME	*MYJOB*			Procedure Name
001 UNIT	*R2*			F1, R1, F2 or R2
010 LOAD NAME	*REPORT*			Columns 75-80 of RPG Control Card or System Program Name
011 UNIT	*R1*			F1, R1, F2 or R2
020 DATE				mmddyy or ddmmyy
030 SWITCH				1-On, 0-Off, X-No Change
040 FILE NAME	*FILE#1*		*FILE#3*	Columns 7-14 of RPG File Description Specifications or Predefined Filename
041 UNIT	*R1*		*F1*	F1, R1, F2 or R2
042 PACK	*VOLA*		*VOLB*	Disk Name (Assigned by Disk Initialization Program)
043 LABEL				VTOC File Name (if different than response to FILE NAME)
044 RECORDS				1-999999 (Maximum Number of Records in File)
045 TRACKS				1-398 (Maximum Number of Tracks for this File)
046 LOCATION				8-405 Location of First Track of File
047 RETAIN				S-Scratch, T-Temporary, P-Permanent
048 DATE				mmddyy or ddmmyy
050 FILE NAME	*FILE#2*		*P/S*	Columns 7-14 of RPG File Description Specifications or Predefined File Name
051 UNIT	*F1*			F1, R1, F2 or R2
052 PACK	*VOLF*			Disk Name (Assigned by Disk Initialization Program)
053 LABEL				VTOC File Name (if different than response to FILE NAME)
054 RECORDS				1-999999 (Maximum Number of Records in File)
055 TRACKS				1-398 (Maximum Number of Tracks for this File)
056 LOCATION				8-405 Location of First Track of File
057 RETAIN				S-Scratch, T-Temporary, P-Permanent
058 DATE				mmddyy or ddmmyy
MODIFY			*RUN*	

Other Possible Entry
(Lines 020-058)
? for Delayed Response

MODIFY OPTIONS

1. Enter RUN
2. Enter CANCEL
3. Correct Statement
   Enter Statement number
   Retype or delete (,) response
4. Create new Statement
   INCLUDE, LOG, FORMS, *(For Comments)

APPENDIX B. GLOSSARY

Activated file — A file whose designation is being changed from scratch to temporary.
Alphabetic — Containing characters A-Z, $, #, and @.
Alphanumeric — Containing both alphabetic and numeric characters.
Chained procedure — Procedures that are connected with an established sequence in which they are to be run.
Compile keywords — Keywords that request information needed to compile a source program.
Compiler — Translates source programs into machine language.
Conversational OCL — OCL for Model 6, called conversational because of the conversation between system and operator.
Data processing system — A network of machine components capable of accepting information, processing it according to a plan, and producing the desired results.
Disk — A physical element of disk storage.
Disk storage — A storage device which uses magnetic recording on flat rotating disks.
End-of-job halt — System halt at the end of every job to give the operator time for any necessary maintenance before beginning the next job.
End-of-statement key — Must be pressed to indicate the end of a response.
Field — In a record, a specified area used for a particular category of data.
File — Group of related records.
File keywords — Keywords that request information needed about a file.
Included statements — Statements you wish to insert in a procedure.
IPL — (Initial Program Load) The process by which the operator loads the program that controls the operation of the system into storage.
Input — Information to be processed.
Job — A piece of work you need done for which a program is written.
Job cycle — The steps involved in carrying out a job.
Job stream — The OCL statements needed for a program.
Keyword — A word, printed by the system, requesting information needed for your program.
Minimum system — System programs necessary to load and run programs.
Numeric — Containing numbers 0 through 9.
Object library — Contains compiled programs, routines, and system programs.
Object library directory — Information concerning each library entry.
Object program — A compiled program stored in the object library.
OCL statement — Consists of a keyword and a response.
Output — Information that has been processed.
Permanent file — Maintained permanently on disk.
Procedure — Sequence of OCL statements in a source library.
Program — Set of instructions written for a job.
Program keywords — Keywords that request information needed about a program.
Prompt — A printed keyword.
Record — Fields grouped together.
Response — A reply to the system's prompted keyword.

Scheduler — Program that provides job-to-job transition.
Scheduler work area — Work area for the Scheduler.
Scratch file — A file used only by the current program which may be overwritten.
Sector — Section of a disk track. There are 24 for each track.
Source library — Contains procedures and source programs.
Source library directory — Information concerning each library entry.
Source statements — Program instructions that have not been compiled.
System directory — Information concerning the libraries and their directories.
Temporary file — A file with short term usefulness which may be overwritten.
Track — Concentric circles on a disk.
Utility programs — Maintenance programs.
VTOC — (volume table of contents) Area on disk containing information about the contents of the disk.

OPERATION CONTROL LANGUAGE

DISK UTILITY PROGRAMS

GC21-7530-0

IBM System/3 Model 6 Printed in U.S.A. GC21-7530-0

IBM

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

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