File No. S360-36
Order No. GC28-6704-4

OS

IBM Systems Reference Library

IBM System/360 Operating System:

Job Control Language Reference

OS Release 21.7

The job control language is used with all System/360 Operating System control programs. Every job submitted for execution by the operating system must include job control language statements. These statements contain information required by the operating system to initiate and control the processing of jobs.

This publication describes the facilities provided with the job control language and contains the information necessary to code job control language statements.

Fifth Edition (August 1976)

This is a reprint of GC28-6704-3 incorporating changes released in the following Technical Newsletter:

GN28-2612 (dated July 23, 1975)

This edition applies to release 21.7 of IBM System/360 Operating System and to all subsequent releases until otherwise indicated in new editions or Technical Newsletters. Changes are continually made to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest *IBM System/360 Bibliography,* GA22-6822, for the editions that are applicable and current.

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

A form for readers' comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Publications Development, Department D58, Building 706-2, PO Box 390, Poughkeepsie, N.Y. 12602. Comments become the property of IBM.

Preface

This publication describes the facilities provided with the job control language and contains the information necessary to code job control language statements. It is intended for use by both the experienced and inexperienced JCL user.

This publication is the result of merging the Job Control Language User's Guide, GC28-6703, into the Job Control Language Reference, GC28-6704. It is a reference book with some introductory material for programmers inexperienced with JCL. All information in this book applies to MFT and MVT configurations of the control program, unless otherwise noted.

This publication consists of five parts preceded by a general introduction:

1. Programming notes, which contain coding conventions used in coding job control language statements.

2. Job control language statements, which describe the format of each statement and the format of the parameters associated with the statement. There is a separate section for each statement.

3. Appendixes, which include additional information on the job control language facilities, such as how to write and use cataloged procedures, and what default values are provided when certain parameters are not coded.

4. Glossary, which contains definitions of many of the terms used in this publication.

5. Foldout charts, which show the format of JOB, EXEC, and DD statement parameters. The foldout charts appear after the index.

Before you read this publication, you should understand the concepts and terminology introduced in the prerequisite publication mentioned below. In addition, the text refers you to other publications for detailed discussions beyond the scope of this publication.

PREREQUISITE PUBLICATION

IBM System/360 Operating System: Introduction, GC28-6534

PUBLICATIONS TO WHICH THE TEXT REFERS

IBM System/360 Operating System:

Data Management for System Programmers, GC28-6550
Utilities, GC28-6586
Operator's Reference, GC28-6691
Operator's Procedures, GC28-6692
Supervisor Services and Macro Instructions, GC28-6646
Data Management Macro Instructions, GC26-3794
MVT Guide, GC28-6720
Storage Estimates, GC28-6551
Tape Labels, GC28-6680
Advanced Checkpoint/Restart, GC28-6708
Programmer's Guide to Debugging, GC28-6670

Contents Directory

Contents

Figures

Summary of Amendments for GC28-6704-3 OS Release 21.7

ISAM Data Sets
- Volume specifications for duplicate DSNAME.

Symbolic Parameters
- Quotes in the PARM field.
- Length cannot exceed 120 characters.

* Parameter
- Addition to rule 4 for syntax checking.
- Use of keywords DLM and DIAGNS.

COND Parameter
- Caution for use of ONLY.

DCB Parameter
- AL and AUL headings assume DCB=OPTCD=Q.
- DEN=0 on 7-track 3240 will result in 556 bpi recording density.
- ASB reader cannot process the DDNAME parameter for instream procedures.
- When specifying PASS, code the DCB parameter or a backward reference to the DCB information.
- DCB=dsname only permitted when DISP=OLD.

DISP Parameter
- Specifying MOD.
- Exclusive control of data set name.
- Specifying CATLG.
- Uncataloging of tape generation data sets that were never opened.
- Reference to Appendix C.

MOD
- Specifying MOD.

SPACE, VOLUME, and UNIT Parameters
- Allocation of space.
- Retain subparameter (volume remains mounted until end of job, except for tape).
- Processing multivolume data sets.
- Volume reference for a backward reference on a DD statement.
- Specifying primary and secondary quantities.
- Cataloged data sets.
- Extending a data set (UNIT parameter).
- Unit affinity and separation.
- Releasing unused space.
- Extending data sets.
- Other minor changes.

Appendix A
- Additional examples.
- Mutually exclusive keywords are allowed during override.

Appendix B
- Retrieving a generation data set.

Appendix C
- How space for index area of an ISAM file is occupied.

Appendix D
- Creating a generation data set.
- The DSNAME parameter.
- Relative generation numbers.

Summary of Amendments for GC28-6704-2 OS Release 21

The Release 21 changes listed below are described in this manual. They are indicated in the text by a vertical line to the left of the change.

DOS Emulator Scheduler Support

New Programming Feature: The DLM parameter is now available on the DD * or DD DATA statement. If the DLM parameter is coded, the delimiter terminating the group of data is the value assigned in the DLM parameter.

DOS/OS Interchange Environment

New Programming Feature: The LTM subparameter of the LABEL parameter allows use of Disc Operating System (DOS) unlabeled tapes with the System/360 Operating System without modifying the tape.

Specification Change: The value "H" coded with the OPTCD parameter under DOS requests the system to check for and bypass any DOS checkpoint records on the tape.

OPEN/CLOSE/EOV Trace Feature

New Programming Feature: The DCB subparameter DIAGNS requests the OPEN/CLOSE/EOV trace option. The trace option gives a module-by-module trace of the OPEN/CLOSE/EOV routines' workarea and the user's DBC.

3505/3525 Card Reader/Punch

New Programming Feature: The FUNC subparameter of the DCB parameter specifies the type of data set to be opened for the 3505/3525 card Read/Punch. The punch unit also interprets the cards punched.

Specification Change: The MODE subparameter of the DCB parameter has been extended to include the 3505/3525 card read/punch.

Reorganization of Publication

This is a combination of topics previously covered in the JCL User's Guide, GC28-6703, and the JCL Reference, GC28-6704, prior to Release 21.

Miscellaneous Changes

New Sections: The "Introduction", the DCB subparameter tables, and the "Summary of the DD Statement" are all new additions to the manual.

Rewritten Sections: The discussion of the COND parameter on the JOB card has been rewritten for increased clarity.

New Devices Included: The following devices are new for Release 21; the 3420/3803 Magnetic Tape Subsystem, the 3410 Magnetic Tape Device, the 3505 Card Reader, the 3525 Card Punch, the 3277 Display Station, the 3284 Printer, and the 3286 Printer.

Glossary: Some additions and amendments have been made to reflect the new and changed material in the publication.

Summary of Amendments for GC28-6704-1 OS Release 20.1

The Release 20.1 changes listed below are described in this manual.

Item	Description
Support for 2305-1, 2305-2, 2319, and 3330	The 2305-1, 2305-2, 2319, and 3330 direct access devices have been added to the section on the UNIT parameter. Z, a character coded in the subparameter OPTCD of the DCB parameter, has an additional meaning when referring to input from a direct access storage device.
Support for 3211	The 3211 printer has been added to the section on the UNIT parameter. FCB, a new parameter to be coded on the DD statement, allows you to specify forms control information. The UCS parameter can also be coded for the 3211; character set codes to be specified in the UCS parameter for the 3211 printer have also been added.
Removal of PCP information	All references to the Primary Control Program have been removed. All information in this manual now applies to systems with MFT or MVT, unless restrictions are specifically noted.

Summary of Amendments for GC28-6704-0 as updated by GN28-2451 OS Release 20

Item	Description
ASCII Support	All references to USASCII have been changed to ASCII (American Standard Code for Information Interchange). In the DCB subparameter BLKSIZE, you can specify the minimum and maximum lengths for blocks of ASCII records on magnetic tape. D and DB can be specified as values for the RECFM subparameter of the DCB parameter; D means that the ASCII records are of variable length and DB means that the ASCII records are of variable length and that they are blocked. A new DCB subparameter BUFOFF allows you to specify a buffer offset for a block of one or more ASCII records on magnetic tape. Q can be specified as a value for the DCB subparameter OPTCD; Q specifies that translation from ASCII input to EBCDIC is required or that translation from EBCDIC to ASCII output is required. AL and AUL are new values for the LABEL parameter; AL specifies that the data set has American National Standard labels and AUL specifies that the data set has both American National Standard labels and American National Standard user labels.
Dynamic Allocation Support for TSO	DYNAM, a new DD statement parameter, allows you to defer definition of a data set until you require it.
NOTIFY Parameter	NOTIFY, a new JOB statement parameter, indicates to the system that you are requesting that a message be sent to your time sharing terminal when your background job completes.
NOPWREAD Subparameter	NOPWREAD, a new subparameter of the LABEL parameter, specifies that a data set can be read without a password, but that the operator must give the password before the data set can be written in or deleted.
TERM Parameter	TERM, a new DD statement parameter, allows you to identify a job as a time-sharing task.
155/165 Model Dependency	3210 and 3215 printer-keyboards have been added to the section on the UNIT parameter.

(Continued)

OS Release 20 (Continued)

Changes to Support TCAM (Telecommunications Access Method)	QNAME, a new parameter for the DD statement, allows you to access messages received by means of TCAM for processing by an application program. Seven new subparameters have been added to the DCB parameter: BUFIN, BUFOUT, BUFMAX, BUFSIZE, PCI, RESERVE, THRESH. In addition, five other subparameters of the DCB parameter may also be used with TCAM: BLKSIZE, BUFL, LRECL, OPTCD, RECFM.
Input/Output Recovery Management Support	The command SWAP has been deleted from the list of commands that can be coded on the command statement.

The Format Of This Publication

This publication is designed for easy reference. The Introduction to this publication contains information that is common to all job control language statements; for instance, one of the topics in this section is how to continue a field onto another control statement. You may want to review the Introduction from time to time.

Section I contains programming notes. This section includes a discussion of format conventions used in this book to describe job control language parameters.

Sections II through X contain descriptions and examples of the different control statements. The job control statements are described in the following order:

II. The JOB statement.

III. The EXEC statement.

IV. The DD statement.

V. The command statement.

VI. The comment statement.

VII. The delimiter statement.

VIII. The null statement.

IX. The PEND statement.

X. The PROC statement.

Each statement description includes the purpose and rules for coding a statement. The JOB, EXEC, and DD statements are described first, in the order in which they normally appear in the input stream. The remaining statements are described in alphabetical order.

The statement description for the JOB, EXEC, and DD statements is followed by a chapter on assigning a name in the name field of the statement and a chapter for each positional and keyword parameter that can be coded on the statement. The chapters on positional parameters appear before the chapters on keyword parameters. Both positional and keyword parameters are described in alphabetical order.

The format of the positional or keyword parameter appears at the beginning of the chapter. Each subparameter is then described briefly. The text following the format description of the parameter describes the purpose of the parameter and each subparameter. Each chapter ends with examples of the use of the parameter and its subparameters.

Section XI consists of Appendixes A through F. These appendixes include:

1. Appendix A: Cataloged and In-stream Procedures

2. Appendix B: Using the Restart Facilities

3. Appendix C: Creating and Retrieving Indexed Sequential Data Sets

4. Appendix D: Creating and Retrieving Generation Data Sets

5. Appendix E: Default Parameter Values Supplied in the Input Reader Procedure

6. Appendix F: A Checklist

Section XII is a glossary of terms used in this publication.

Section XIII, which follows the index, is a set of foldout charts. These charts summarize syntax of JOB, EXEC, and DD statement parameters.

Introduction to the Job Control Language

The IBM System/360 Operating System was designed to meet the many diverse needs of the computer user. Data processing needs differ greatly from one installation to another and between individual users within an installation. The purpose of the operating system is to aid you in getting your work done. It achieves its purpose by managing all available resources, including the central processing unit, main storage, input/output devices, and any programs that are a part of the system. To use the operating system, you must describe to the system the work you want done and the resources you will need. You provide the operating system with this information through use of the job control language.

The job control language, commonly referred to as JCL, consists of nine control statements. On these control statements, you code information to be used by the operating system to direct the execution of the programs you have written. The programs you write are called source programs (or source modules). Your source program and the JCL statements needed to describe what the operating system is to do for you, together with whatever related data you may have, constitute a job. Every job submitted for execution by the operating system must include JCL statements. The design and coding of the JCL portion of your job can require a considerable amount of time.

The job control language is a very flexible language and with this flexibility come many optional features. You should become familiar enough with the language to be able to decide what information the operating system will need to process your job and which features of the language will aid you in getting your job done most efficiently.

The IBM System/360 Operating System

An IBM System/360 Operating System consists of a control program together with a number of optional processing programs such as language translators, utility programs, and a sort/merge program. The purpose of the control program is to efficiently schedule, initiate, and supervise the work performed by the computing system. The processing programs (see figure 1) are designed to help you program solutions to problems and design new applications. They do this by giving you a combination of programming aids, services, and precoded routines that you can use with whatever programming language you choose.

PROCESSING PROGRAMS AND JCL

You can use the processing programs provided by IBM singly or in combination to process your job. The IBM processing programs available for your use can be supplemented with programs written by you or others at your installation. The IBM processing programs are classified as either language translators or service programs (see figure 1).

The language translators enable you to write a problem solution or an application in a language that can be more readily learned and more easily used than the strictly numerical machine language of the computing system. IBM provides six language translators (see Figure 1). The language translators create machine language programs based upon computer programs written in higher-level languages. All higer-level language translators are called compilers. The assembler is a low-level language translator. That is, each assembler statement translates into

one machine instruction. A compiler generates one or more machine instructions for each higher-level language statement. This process is known as compilation (or assembly in the case of the assembler). The machine language program that is produced from a source program is called an object module.

Processing Programs	
Language Translators	Service Programs
ALGOL Assembler COBOL FORTRAN PL/I RPG	Linkage Editor Loader Sort/Merge TESTRAN Data set utilities System utilities Independent utilities

Figure 1. Processing Programs

The linkage editor is one of the service programs. It combines object modules that have been individually compiled or assembled. The result is a load module. A load module is one ready to be loaded into main storage and executed. Another service program you can use to accomplish link editing is the loader. It combines linkage editing and execution by loading object modules produced by the language translators and load modules produced by the linkage editor into main storage for execution. Other service programs supplied to aid in processing jobs are the sort/merge program and the utility programs. The sort/merge program is a generalized program that can be used to sort or merge fixed- or variable-length records in ascending or descending order. The utility programs are divided into three subsets: data set, system, and independent utilities. Data set utilities are designed to help you in the manipulation of data. They aid you in doing such things as transferring, copying, or merging sets of data from one I/O device onto another. The system utilities are used to change or extend the indexing structure of the system library catalog and to print an inventory of the data and programs that are cataloged in the system library. Independent utilities are used chiefly by the system programmer to prepare direct access storage for use under the operating system.

In order to use the processing programs you simply request the particular program you want by coding the name of the program on a job control language statement. For example, you may write a program in COBOL to process insurance premium payments. Your program must be compiled (translated into machine language) and linkage edited before it can be executed. This means that your job will be organized into three parts. The parts of a job are known as job steps and, in this case, you would have three job steps.

In the first step, you code the name of the COBOL compiler you are requesting on a JCL statement. In this step, you also include JCL statements to describe any data sets that the compiler requires. The COBOL compiler will translate your source program into machine instructions and produce an object module. In the next step, you use a JCL statement to request the linkage editor. Again you include JCL statements to describe any data sets that may be required by the linkage editor. The linkage editor uses the object module as its input data and produces a load module. A load module is the executable form of a program. In the last step, you request that your program (the load

module form) be executed. You will have to describe any data sets that will be used by your program (such as the actual insurance premiums and the master file) and where the output of the job step is to go.

THE CONTROL PROGRAM AND JCL

The control program must perform three functions: job management, task management, and data management.

- Job management involves reading and interpreting job control language statements, scheduling jobs, initiating and terminating jobs and job steps, and recording job output data.
- Task management monitors and controls the entire operating system, and is used throughout the operation of both the control program and processing programs.
- Data management's purpose is to simplify storage, retrieval, and maintenance of all data, regardless of the way it is organized.

Through the use of job control language (JCL), you communicate with the job management area of the control program and specifically with the job scheduler. Figure 2 gives you a brief summary of the components of job management and a synopsis of what each component does. JCL statements indicate to the job scheduler the work you want done. With JCL statements, you tell the job scheduler at what point your job begins, the name of your job, how you organized your job, where your data is, the programs you want executed, and the main storage requirements of those programs.

MASTER SCHEDULER	JOB SCHEDULER
• Relays messages to and from the system to or from the operator. • Executes operator commands. • Responds to replies from the operator. • Starts and stops the reader/interpreter, initiator/terminator, and the output writer tasks.	<u>Reader/Interpreter</u> • Reads and analyzes job control Statements from the input stream. • Places information contained in the job control statements into a series of tables. Initiator/Terminator • Allocates resources required to perform a step of the job. • Loads and transfers control to the program that is to be executed to perform the job step. • Terminates the job step when execution of the program is completed. • Selects a job from the input work queue. <u>Output Writer</u> • Controls the writing of job output data.

Figure 2. Job Management

The job scheduler consists of three areas: the reader/interpreter, the initiator/terminator, and the output writer. The reader/interpreter reads and analyzes your job control statements. It checks to make sure that you have not made errors in coding your statements. The reader/interpreter places the information contained in the job control statements into a series of tables for system use. The initiator/terminator assigns to each step of your job the resources that you have requested, notifying the operator of any tapes or disk packs that have to be mounted. A job step is a logical division of your job. Each step is associated with one processing program or procedure, and related data. A job consists of one or more job steps.

After the initiator/terminator has assigned the requested resources to a job step, it requests the supervisor program, a part of task management, to initiate the execution of the program you have specified in your job step. When the program is completed, the initiator/terminator terminates the job step, releasing the resources assigned to the step and, thereby, making them available for use by other job steps. After the execution of the last step in your job, the output writer records the output of your job. Your output is directed to the device you have specified in your JCL.

Control Program Configurations

There are two configurations of the control program:

- MFT -- multiprogramming with a fixed number of tasks.
- MVT -- multiprogramming with a variable number of tasks.

The MFT control program reads jobs in sequential order from up to three input streams concurrently. Up to 15 job steps, from 15 different jobs, can be performed simultaneously. The MFT control program can also concurrently record as many as 36 streams of job output.

The MVT control program reads one or more input streams of jobs and schedules the jobs in order of priority (you assign priority). Up to 15 independent jobs can be performed concurrently. Job steps within a single job are performed in sequential order because one step may depend on the completion of another. Within a job step, any number and type of data processing tasks can be initiated. The MVT control program can concurrently record job system output on as many as 36 devices.

The job control language statements are basically the same for the two configurations, but some parameters coded on the statements are not meaningful for both. For example, in both MFT and MVT, you can use a parameter to assign a dispatching priority to a job. In MVT, you can also use a parameter to assign a dispatching priority to a job step. This parameter, however, has no meaning in MFT. All the parameters in this book apply to systems with MFT and MVT unless otherwise noted.

Job Control Language Statements

The nine job control language statements used to describe a job to the system are:

1. Job (JOB) statement.
2. Execute (EXEC) statement.
3. Data definition (DD) statement.

4. Delimiter statement.

5. Null statement.

6. Procedure (PROC) statement.

7. Procedure end (PEND) statement.

8. Comment statement.

9. Command statement.

A job control statement consists of one or more 80-byte records. Most jobs are submitted to the operating system for execution in the form of 80-column punched cards or as card images off direct access devices. The operating system is able to distinguish a job control statement from data included in the input stream. In columns 1 and 2 of all the statements except the delimiter statement, you code //. For the delimiter statement, you code /* in columns 1 and 2 and this notifies the operating system that the statement is a delimiter statement. For a comment statement, you code //* in columns 1, 2, and 3 respectively.

Parameters coded on these JCL statements help the job scheduler to regulate the execution of jobs and job steps, retrieve and dispose of data, allocate I/O resources, and communicate with the operator.

JOB STATEMENT

The job statement (or JOB statement) indicates to the system at what point a job begins (see figure 3). On the JOB statement, you code the name of your job. This name is used to identify messages to the operator and to identify your program output. By using the parameters allowed on the JOB statement, you can provide accounting information for your installation's accounting routines, specify conditions for early termination of your job, assign job priority, request a specific class for job scheduler messages, hold a job for later execution, and limit the maximum amount of time the job may use the central processing unit (CPU). With MVT, you can also specify the amount of main storage to be allocated to the job.

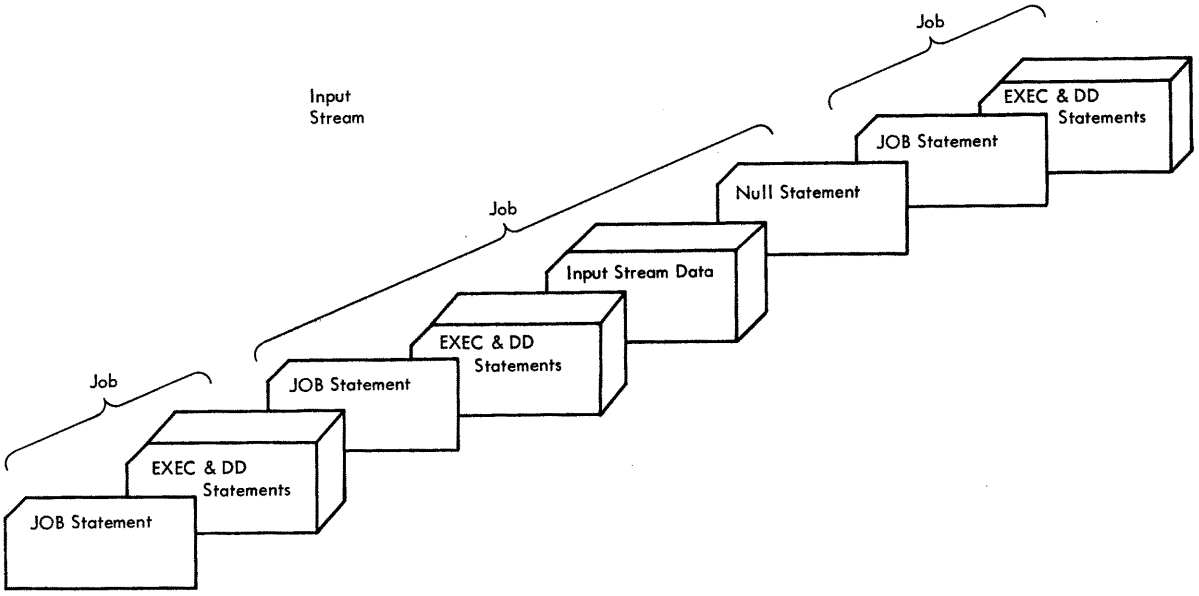


Figure 3. Defining Job Boundaries

EXEC STATEMENT

The EXEC statement marks the beginning of a job step and the end of the preceding step (see figure 4). On the EXEC statement, you identify the program to be executed or the cataloged procedure or in-stream procedure to be called. A cataloged procedure is a set of job control language statements that has been assigned a name and placed in a partitioned data set known as the procedure library.

The EXEC statement can also be used to provide job step accounting information, to give conditions for bypassing or executing a job step, to assign a limit on the CPU time used by a job step, and to pass information to a processing program such as the linkage editor. All this information is communicated to the system by the parameters that you can code on the EXEC statement. In systems with MVT, you can use a parameter to specify the amount of main storage to be allocated to the job step.

DD STATEMENT

A DD statement identifies a data set and describes its attributes. There must be a DD statement for each data set used or created in a job step. The DD statements are placed after the EXEC statement for the step. The parameters of the DD statement provide the system with such information as the name of the data set, the name of the volume on which it resides, the type of I/O device that holds the data set, the format of the records in the data set, whether a data set is old or new, the size of newly created data sets, and the method that will be used to create or access the data set. The name of the DD statement provides a symbolic link between a data set (on data file) named in your program and the actual name and location of the corresponding data set. This symbolic link allows you to relate the data set in your program to different data sets on different occasions.

DELIMITER AND NULL STATEMENTS

The delimiter statement (or /* statement) and null statement (or // statement) are markers in an input stream. The delimiter statement is used to separate data placed in the input stream from any JCL statement that may follow the data. The null statement can be used to mark the end of the JCL statements and data for a job.

PROC AND PEND STATEMENTS

The PROC statement may appear as the first JCL statement in a cataloged or in-stream procedure. For cataloged procedures or in-stream procedures, the PROC statement is used to assign default values to parameters defined in a procedure. An in-stream procedure is a set of job control language statements that appear in the input stream. The PROC statement is used to mark the beginning of an in-stream procedure. The PEND statement is used to mark the end of an in-stream procedure.

COMMENT STATEMENT

The comment statement can be inserted before or after any JCL statement that follows the JOB statement and can contain any information you think would be helpful to you or anyone interested in your program.

COMMAND STATEMENT

The command statement is used to enter commands through the input stream. Commands can activate and deactivate system input and output units, request printouts and displays, and perform a number of other operator functions.

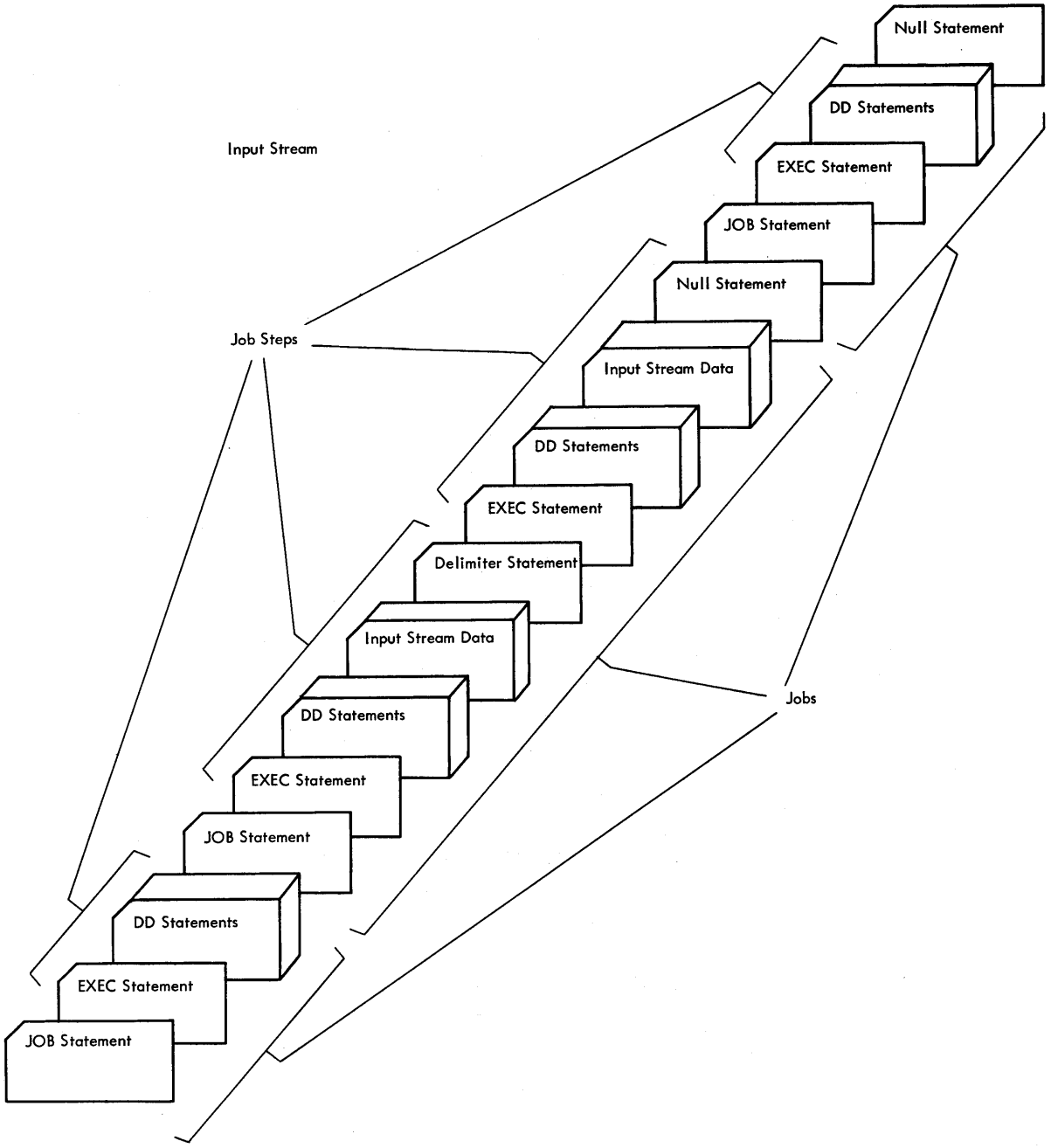


Figure 4. Defining Job Step Boundaries

Defining Your Job

Now that you have been introduced to the nine JCL statements, let us use these statements to define a job. Basically, the statements with which you will be most concerned are the JOB, EXEC, the DD statements. The delimiter and null statements may be used but they are usually unnecessary. That is, the system will provide delimiters by default at the end of the data set.

Assume you have coded and punched (transcribed onto cards) a program to process payroll records. The program is coded in PL/I and, like the insurance premium program discussed earlier, it must be compiled and link-edited (or linkage edited) before it can be executed. Therefore, your job will have three steps: compilation, link-editing, and execution.

First, you must code a JOB statement to mark the beginning of your job. On this statement, you must assign a name to your job so that both the system and the operator will be able to identify it. For example, you could code:

```
//PAYROLL     JOB  (D58706,GROUP1),WOODEN

  Name of          Accounting     Programmer's
  the job          information    name
```

After the JOB statement, you code an EXEC statement to mark the beginning of the first step. On this statement, you code the name of the PL/I compiler you are requesting to translate your source program into machine language. For example, you would code:

```
//COMPILE     EXEC      PGM=IEMAA,PARM='LOAD,NODECK'

  Name of              Name of the          Information being
  the job step         PL/I compiler        passed to the compiler
                       requested
```

Following the EXEC statement, you code a DD (data definition) statement for each data set the PL/I compiler requires. One of these DD statements must tell the compiler that you are placing your source program in the input stream. Another DD statement must be used to tell the compiler where to place the machine language translation of your program (the object module). The other DD statements should be used to define work areas, for printing messages, and for listings. (A listing is a printout of the source language statements of a program.) After you code all the DD statements needed by the PL/I compiler, you code a delimiter statement to separate your source program from the JCL statements of the next step. Your source program will be placed in the input stream immediately following a //SYSIN DD * statement and before the delimiter statement.

The next step begins with an EXEC statement. On this statement, you request execution of the linkage editor. The linkage editor, like the PL/I compiler, will require certain data sets. Each data set required by this step must have a DD statement coded to describe it. In this step, you must include a DD statement telling the linkage editor where the PL/I compiler placed the object module. You must also include a DD statement telling the linkage editor where it is to place the load module it produces. The load module is the executable form of your program. Once it is loaded into main storage, it can be executed by the central processing unit. Other DD statements should be included for work areas, for printing messages, and for listings.

The third step calls for the actual execution of your program. The EXEC statement for this step requests that your program (the load module produced by the previous step) be executed. You will need to code DD statements to tell your program where the payroll records are that you intend to process. Following the DD statements for this step, you can include the payroll records (if they are in card form). Otherwise you indicate to your program that the payroll records are on magnetic tape or direct access. The last DD statement for the step should indicate to the system that the data following it is to be used by your program. You will need DD statements to tell your program where the master file is and where you want the output of your program to go, and to define any work areas or other data required by your program. You can code a null statement to be placed at the end of your deck. The null statement will indicate to the system that this is the end of your job. Once your entire job (the JCL, the payroll records, and your source program) is either properly identified by JCL control cards or included in the input stream (see figure 3), it is ready to be submitted for processing.

Another way of defining your job is through the use of a cataloged procedure. IBM supplies a catalog procedure which defines the steps needed for compilation, link-editing, and executing your program, or you can write your own. The procedure described in the previous discussion might be cataloged for this purpose.

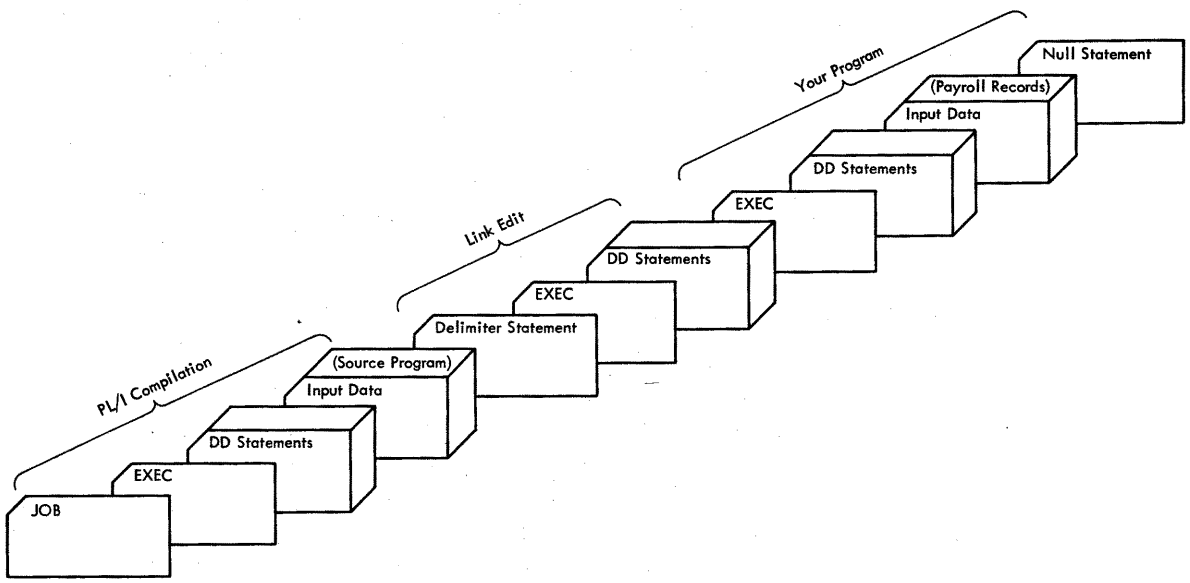


Figure 5. Your Job

Cataloged and In-Stream Procedures

Often the same set of job control statements is used repeatedly with little or no change (for example, to specify compilation, link-editing, and execution of programs). To save programming time and to reduce the possibility of error, standard job step definitions can be prepared and placed (or cataloged) in a partitioned data set known as the procedure library. The procedure library (SYS1.PROCLIB) is a system data set maintained on direct access storage by the control program. A set of job control language statements placed in the procedure library is called a cataloged procedure. A cataloged procedure consists of EXEC and DD statements.

By simply using a JOB statement and an EXEC statement, you can retrieve a specific catalog procedure. You specify on the EXEC statement the name of the procedure you want. This directs the job scheduler to use the job step definition from the procedure library. The effect is the same as if the job control statements of the cataloged procedure appeared in the input stream in the place of the EXEC statement that calls the procedure. If necessary, you can modify the cataloged procedure by a process known as overriding.

Before putting a procedure into the procedure library, you may want to test it. This can be done by converting the procedure to an in-stream procedure. An in-stream procedure is a set of JCL statements placed in the input stream that can be used any number of times during a job by naming that procedure in an execute (EXEC) statement. Another advantage to in-stream procedures is that they can give you the facility of a cataloged procedure without being placed on the procedure library. After testing the procedure, you may keep it in card form and simply insert it in the input stream whenever you want to use it.

Processing Your Job

To have a job processed, you must submit the JCL statements and any related input data to the operating system through an input/output (I/O) device chosen by the operator. The input unit can be a card reader, a magnetic tape, a telecommunications line, or a direct access device. The sequence of JCL statements and input data for all the jobs being submitted through an input unit is called the input stream.

Assume you submit a PL/I payroll job to be processed by a system with MFT. The program is in the form of punched cards, called a deck. The operator places your deck in the card reader (input unit) together with decks for other jobs to be processed. In this case, the card decks for all these jobs constitute the input stream (see figure 4).

The operator starts the system reader; that is, he instructs the operating system (job management) to start reading the input stream. Job management stores the job control statements in the job queue data set (SYS1.SYSJOBQE) until they are used. Then it examines the first step and determines its needs. The first step of your job requests the PL/I compiler and defines several data sets. The operating system (data management) determines whether there is any space available on the devices you requested for the data sets the PL/I compiler will create during this step (for example, the object module) and whether the data sets required by the compiler are available (for example, your source program).

If all data set requirements are met, the PL/I compiler is brought into main storage and given control. After your program is compiled, the operating system reads and determines the requirements of the second step which requests the linkage editor. The operating system performs the same operations for the data sets required by the linkage editor and

then brings the linkage editor into main storage and gives it control. After the linkage editor produces the load module, the operating system reads and processes the next step. Its requirements are determined, and your program is brought into main storage and given control.

While this job is being executed, the system can also execute up to 15 other jobs from up to 3 input streams.

Capabilities of the Job Control Language

The job control language provides you with many capabilities to help in efficiently getting your job coded and processed. The language allows you to:

- Specify the device requirements of a program at the time it is executed rather than when it is assembled or compiled. You do this by writing a program in such a way that it is not directly tied to a particular I/O device. A device-independent program could, for example, accept an input data set from any magnetic tape or direct access device, or from any card reader; output could be recorded on any appropriate I/O device. At the time you submit the program for execution, you code in your JCL the type of device required.

- Copy existing data set names, control statements, and control blocks with a backward reference facility to reduce recoding. When coding a DD statement, you simply use this facility to refer the system to an earlier DD statement that contains certain information you want copied.

- Pass data sets used by more than one step from one step to another, to reduce mounting and retrieval time.

- Retrieve a data set by name using the system catalog, eliminating the need to know its exact location.

- Optimize use of channels, units, volumes, and direct access space. For example, when two or more data sets are to be used in a job step, processing time may be shortened by requesting that the system transmit data over separate channels (A channel is a hardware device that connects a CPU and main storage with input/output control units). It would be faster to have your input data set and your output data set on separate channels than to have them on the same channel. A JCL parameter allows you to request channel separation for data sets in each job step.

- Specify that data sets are to be shared by two or more job steps that are operating independently.

- Classify jobs according to their characteristics and importance so that the system may balance the mix of jobs for more efficient operation. The characteristics of the job will determine its class and the turnaround time required by a job will determine its priority. For example, an installation may assign jobs that use a large amount of main storage to one class, jobs that run for a long time to another, and teleprocessing jobs to another class. Within each class you may assign priorities to determine the order of execution. In the class of "jobs that run for a long time", you may wish to assign a higher priority to the weekly payroll program than to the monthly analysis program. Each job is executed one step at a time and steps of different jobs can be interleaved. For example, if, while the system is executing a job that runs for a long time, enough resources are available to process teleprocessing jobs, several teleprocessing jobs can also run.

Once you learn the basics of the job control language, you should become familiar with these and other capabilities of the language that have been designed to make the most efficient use of the operating system. Part II of this publication will introduce you to the various parameters that can be coded on the JCL statements. The facilities that have been briefly explained here are discussed in greater detail with examples of their use.

Section I: Programming Notes

The formats of the parameters described in this publication for the JOB, EXEC, and DD statements appear at the beginning of the chapter on the corresponding parameter. Notations used in the format descriptions are described below.

1. Uppercase letters and words are coded on the control statement exactly as they appear in the format description, as are the following characters.

ampersand	&
asterisk	*
comma	,
equal sign	=
parentheses	()
period	.

2. Lowercase letters, words, and symbols appearing in the format description represent variables for which specific information is substituted when the parameter is coded.

   For example, PRTY=priority is the format description for the PRTY parameter. When you code the PRTY parameter on a JOB statement, you substitute a number for the word "priority."

3. Braces { } are a special notation and are never coded on a control statement. Braces are used to group related items; they indicate that you must code one of the items.

   For example, {TRK / CYL / block size} is part of the format description for the SPACE parameter. When you code the SPACE parameter, you must code either TRK, CYL, or a substitute for "block size," which would be a number.

4. Brackets [ ] are a special notation and are never coded on a control statement. Brackets indicate that the enclosed item or items are optional and you can code one or none of the items.

   For example, [,DEFER] is part of the format description for the UNIT parameter. When you code the UNIT parameter, you can include ,DEFER in the UNIT parameter or omit it.

   An example of more than one item enclosed in brackets is

   [EXPDT=yyddd / RETPD=nnnn], which is part of the format description for the LABEL parameter. When you code the LABEL parameter, you can include either EXPDT=yyddd or RETPD=nnnn in the LABEL parameter or omit both.

   Sometimes, one of a group of items enclosed in brackets is a comma. You code the comma when none of the other items in the group is used and a following part of the parameter is still to be coded.

The comma indicates to the system that you have not selected to code any of the items enclosed in the brackets.

For example, [,progname][,form number]) is part of the format description for the SYSOUT parameter. When you code the SYSOUT parameter, you have the option of coding both ",progname" and ",form number", omitting both, or coding only one. The comma enclosed in brackets with ",progname" must be coded when ",progname" is not coded but ",form number" is coded; that is, you would code: ,,form number).

5. An ellipsis ... (three consecutive periods) is a special notation and is never coded on a control statement. An ellipsis is used to indicate that the preceding item can be coded more than once in succession.

   For example, COND=((code,operator),...) is the format description for the COND parameter on the JOB statement. The ellipsis indicates that (code,operator) can be repeated.

Fields in Control Statements

Every control statement is logically divided into different fields. There are four fields -- name field, operation field, operand field, comments field -- but not all of the control statements can contain all of these fields. Figure 6 shows the fields for each statement.

Statement	Columns 1 and 2	Fields
Job	//	name operation(JOB) operand[1] comments[1]
Execute	//	name[1] operation(EXEC) operand comments[1]
Data Definition	//	name[1] operation(DD) operand comments[1]
PROC(Cataloged)	//	name[1] operation(PROC) operand comments[1]
PROC(in-stream)	//	name operation (PROC) operand[1] comments[2]
Procedure end	//	name[1] operation(PEND) comments[1]
Command	//	operation(command) operand comments[1]
Delimiter	/*	comments[1]
Null	//	

Statement	Columns 1,2,3	Field
Comment	//*	comments

[1]Optional
[2]Optional -- If operand(s) are not coded, comments cannot be coded. If operand(s) are coded, comments are optional.

Figure 6. Control Statement Fields

The name field identifies the control statement so that other statements and system control blocks can refer to it. The name field is 1 to 8 alphameric and national (#, @, $) characters; the first character must be alphabetic or national. The name field must begin in column 3.

The operation field specifies the type of control statement, or, in the case of the command statement, the command. The operation field must follow the name field and must be preceded and followed by at least one blank.

The operand field contains parameters separated by commas. The operand field must follow the operation field and must be preceded and followed by at least one blank. The operand field is described in more detail in the next chapter "Parameters in the Operand Field."

The comments field contains any information deemed helpful by the person who codes the control statement. The comments field must follow the operand field and must be preceded by at least one blank.

Control statement fields -- except the name field, which must begin in column 3 -- can be coded in free form. Free form means that the fields need not begin in a particular column. Separate each field with a blank; the blank serves as a delimiter between fields.

Except for the comment statement, which can be coded through column 80, fields cannot be coded past column 71. If the total length of the fields will exceed 71 columns, you must continue the fields onto one or more succeeding statements. How to continue fields is described in the chapter "Continuing Control Statements."

Some examples of how the different fields appear on control statements are:

Columns:
1 2 3

Name	Operation	Operand	Comments
/ / JOB8	JOB	MSGLEVEL=(1,1)	THE FIRST STATEMENT IN JOB
/ / STP1	EXEC	PGM=PROG4,REGION=80K	EXECUTES PROGRAM NAMED PROG4
/ / WORK	DD	UNIT=2400	DEFINES A TEMPORARY DATA SET

Parameters in the Operand Field

The operand field is made up of two types of parameters: one type is characterized by its position in the operand field in relation to other parameters (a positional parameter); the other type is positionally independent with respect to others of its type, and is characterized by a keyword followed by an equal sign and variable information (a keyword parameter). Both positional parameters and the variable information associated with keyword parameters can assume the form of a list of several items (subparameters) of information.

All positional and keyword parameters and subparameters coded in the operand field must be separated from one another by commas.

Positional parameters must be coded first in the operand field in a specific order. The absence of a positional parameter is indicated by a comma coded in its place. However, if the absent parameter is the last one, or if all later positional parameters are also absent, you need not code replacing commas. If all positional parameters are absent from the operand field, you need not code any replacing commas.

Keyword parameters can be used anywhere in the operand field with respect to one another. Because of this positional independence, you need not indicate the absence of a keyword parameter.

A positional parameter or the variable information in a keyword parameter sometimes assumes the form of a list of subparameters. Such a list may be composed of both positional and keyword subparameters that follow the same rules and restrictions as positional and keyword parameters. You must enclose a subparameter list in parentheses, unless the list reduces to a single subparameter.

The EXEC statements and DD statements in cataloged procedures can contain one other type of parameter -- a symbolic parameter. A symbolic parameter is characterized by a name preceded by an ampersand (&); a symbolic parameter stands as a symbol for a parameter, a subparameter, or a value. Symbolic parameters allow you to make any information in the operand field of a procedure EXEC statement or DD statement variable. A value to be assumed by a symbolic parameter may be coded on the EXEC statement that calls the procedure. This value is in effect only while the procedure is being executed. For a detailed discussion on how to assign values to symbolic parameters, refer to the chapter "Assigning Values to Symbolic Parameters" in Appendix A; for a detailed discussion on how to use symbolic parameters in a set of control statements that you plan to catalog as a procedure, refer to the chapter "Using Symbolic Parameters in a Procedure" in Appendix A.

Continuing Control Statements

When the total length of the fields on a control statement will exceed 71 columns, you must continue the fields onto one or more succeeding statements.

The command, comment, delimiter, and null statements cannot be continued.

You can continue the operand field or the comments field. To continue either of these fields, you must follow the continuation conventions.

To continue the operand field:

1. Interrupt the field after a complete parameter or subparameter, including the comma that follows it, at or before column 71.

```
1-10      11-20     21-30     31-40     41-50     51-60     61-70     71-80
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
//DD4 DD  DSNAME=PROB43881,DISP=(NEW,KEEP,DELETE),
```

2. Comments can be included by following the interrupted field with at least one blank.

```
1-10      11-20     21-30     31-40     41-50     51-60     61-70     71-80
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
//DD4 DD  DSNAME=PROB43881,DISP=(NEW,KEEP,DELETE), DATA SET FOR TEST81
```

3. Optionally, code any nonblank character in column 72. (The nonblank character in column 72 is required only when you are continuing a comments field.) If you do not code a character in column 72 when continuing the operand field, the system treats the next statement as a continuation statement as long as you follow the conventions outlined in items 4 and 5.

```
1-10      11-20     21-30     31-40     41-50     51-60     61-70     71-80
//DDH1 DD  DSNAME=PROB43.B.D1.SP=(NEW,KEEP,DELETE),  DATA SET FOR TEST 1  X
```

4. Code the identifying characters // in columns 1 and 2 of the following statement.

```
1-10      11-20     21-30     31-40     41-50     51-60     61-70     71-80
//
```

5. Continue the interrupted operand beginning in any column from 4 through 16. If you leave the statement blank after column 2 or if you begin coding after column 16, the system assumes that no other operands are present and treats any characters you code as a comment field.

```
1-10      11-20     21-30     31-40     41-50     51-60     61-70     71-80
//  UNIT=2311,VOLUME=SER=BT3T84,SPACE=(800,(150,25))
```

To continue the comments field:

1. Interrupt the comment at a convenient place before column 72.

```
1-10      11-20     21-30     31-40     41-50     51-60     61-70     71-80
//STEP1   EXEC PGM=A81.9,REGION.M=86K,RD=R,DPRTY=(13,12)  RESULTS OF TEST
```

2. Code a nonblank character in column 72.

```
1-10      11-20     21-30     31-40     41-50     51-60     61-70     71-80
//STEP1   EXEC PGM=A81.9,REGION.M=86K,RD=R,DPRTY=(13,12)  RESULTS OF TEST X
```

3. Code the identifying characters // in columns 1 and 2 of the following statement.

```
1-10      11-20     21-30     31-40     41-50     51-60     61-70     71-80
//
```

4. Continue the comments field beginning in any column after column 3.

```
1-10      11-20     21-30     31-40     41-50     51-60     61-70     71-80
//        SHOULD BE SENT TO MR. ECKERT
```

Any control statements in the input stream, other than a comment statement, that the system considers to contain only comments have //*

in columns 1 through 3 on an output listing. Any control statements in a cataloged procedure, other than a comment statement, that the system considers to contain only comments have XX* in columns 1 through 3 on an output listing. For an in-stream procedure +++ appears in columns 1-3. In both cases for a comment statement, *** appears in columns 1 through 3 on an output listing.

Backward References

A facility of the job control language allows you to refer the system to an earlier DD statement in the job for certain information. A backward reference is of the following form:

- parameter=*.ddname -- use this form when the earlier DD statement is contained in the same job step.
- parameter=*.stepname.ddname -- use this form when the earlier DD statement is contained in an earlier job step.
- parameter=*.stepname.procstepname.ddname -- use this form when the earlier DD statement is contained in a cataloged procedure called by an earlier job step. ("Stepname" is the name of the step that calls the procedure.)

You can use the backward reference facility only with certain parameters. These parameters and the information the system obtains when the backward reference facility is used are:

- PGM -- the data set that contains the program to be executed in this job step.
- DCB -- all DCB subparameters coded on the earlier DD statement. (If you code any DCB keyword subparameters following the backward reference, these subparameters override any of the corresponding subparameters coded on the earlier DD statement. If a DD statement defines an existing data set and contains a backward reference in the DCB parameter, the system copies only those subparameters from the earlier DD statement that were not previously specified for the existing data set.)
- DSNAME -- the name of the data set being defined on this DD statement.
- VOLUME-REF -- the volume serial number(s) on which the data set resides or will reside; unit information is also obtained by the system.

Concatenating Data Sets

Up to 255 sequential or up to 16 partitioned input data sets, each of which may reside on a different volume, can be logically connnected for the duration of a job step. To concatenate data sets, simply omit the ddnames from all the DD statements except the first in the sequence. When this ddname is encountered in a data control block in the processing program, each data set is automatically processed, in the same sequence as the DD statements defining them.

If concatenated data sets have unlike characteristics, e.g., the device types, block lengths, or record formats differ, the DCBOFLGS field of the data control block must be modified while the program is executing. For details, refer to the topic "Concatenating Sequential and Partitioned Data Sets" in the Data Management Services publication.

If you make a backward reference to a concatenation (using an asterisk), the system obtains information only from the first data set defined in the sequence.

If you make a forward reference to a concatenation (using the DDNAME parameter), the system only obtains information from the first data set defined in the sequence.

You should not concatenate other data sets to a data set you have defined using the DUMMY parameter. When the processing program asks to read a dummy data set, an end-of-data-set exit is taken immediately and any concatenated data set is ignored.

The following example illustrates a group of DD statements defining concatenated data sets, including a data set in the input stream.

```
//INPUT  DD   DSNAME=A.B.C,DISP=(OLD,DELETE)
//       DD   DSNAME=X.Y.Z,DISP=OLD,LABEL=(,NL)
//       DD   DSNAME=ALPHA,UNIT=2311,VOLUME=SER=P12,DISP=(OLD,DELETE)
//       DD   *
              .
              .
              .
             data
              .
              .
              .
/*
```

Character Sets

Job control statements are coded using a combination of the characters from three different character sets. The contents of each of the character sets are described in figure 7.

Character Set	Contents	
Alphameric	Alphabetic Numeric	A through Z 0 through 9
National	"At" sign Dollar sign Pound sign	@ $ #
Special	Comma Period Slash Apostrophe Left parenthesis Right parenthesis Asterisk Ampersand Plus sign Hyphen Equal sign Blank	, . / ' ( ) * & + - =

Figure 7. Character Sets

When you code any special characters, certain rules must be followed. These rules and the use of special characters are described next.

Using Special Characters

Special characters are used in the job control language to:

1. Delimit parameters (the comma).

2. Delimit fields (the blank).

3. Perform syntactical functions. (For example, the appearance of && as the first two characters following DSNAME= tells the system that a temporary data set name follows. The appearance of / in the UNIT parameter, UNIT=293/5, tells the system that a specific 2321 bin is desired.)

Sometimes you can code a special character that does not satisfy one of the three uses of special characters. In most of these cases, you must indicate that special characters are being used by enclosing the item that contains the special characters in apostrophes (5-8 punch), e.g., ACCT='123+456'. If one of the special characters is an apostrophe, you must code two consecutive apostrophes (two 5-8 punches) in its place, e.g., 'O'NEILL'.

The following list contains those parameters that can have special characters as part of their variable information, and indicates when the apostrophes are not required.

1. The accounting information on the JOB statement. The account number and additional accounting information can contain hyphens without being enclosed in apostrophes.

2. The programmer's name on the JOB statement. The programmer's name can contain periods without being enclosed in apostrophes.

3. The checkid field in the RESTART parameter on the JOB statement.

4. The ACCT parameter on the EXEC statement. The ACCT parameter can contain hyphens without being enclosed in apostrophes.

5. The PARM parameter on the EXEC statement.

6. The DSNAME parameter on the DD statement. The DSNAME parameter can contain hyphens without being enclosed in apostrophes. If the DSNAME parameter contains a qualified name, it can contain periods without being enclosed in apostrophes. If the DD statement identifies a generation of a generation data group, the generation number in the DSNAME parameter can contain a plus or minus (hyphen) sign without being enclosed in apostrophes. If the DD statement defines a temporary data set, the DSNAME parameter can contain, as the first two characters, ampersands without being enclosed in apostrophes. If the DD statement defines a member of a partitioned data set, a generation of a generation data group, or an area of an indexed sequential data set, the DSNAME parameter contains parentheses that enclose the member name, generation number, or area name; these parentheses are not enclosed in apostrophes.

7. The volume serial number in the VOLUME parameter on the DD statement. The volume serial number can contain hyphens without being enclosed in apostrophes.

8. The DLM parameter on the DD statement.

Coding Form

For your convenience in coding control statements, you can use Form N74167, a punch card containing formatted lines, each representing a different type of statement. (See figure 8.) Some of the lines can be used for concatenations, overrides, and continuation statements.

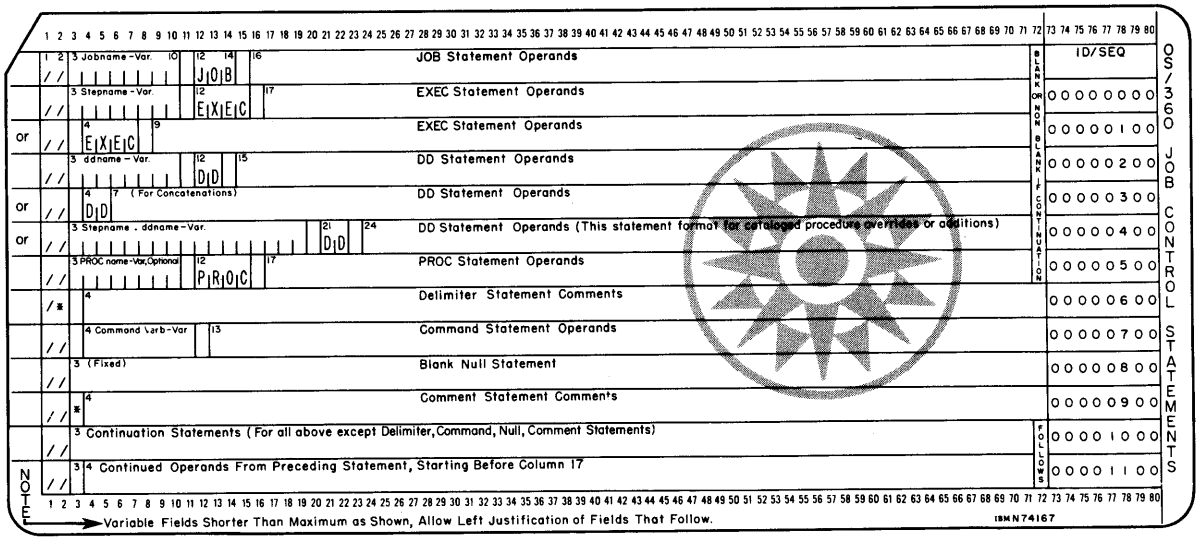


Figure 8. Coding Form for Coding Control Statements

Section II: The JOB Statement

The JOB statement marks the beginning of a job and, when jobs are stacked in the input stream, marks the end of the control statements for the preceding job. The JOB statement must contain a valid jobname in its name field. All parameters in its operand field are optional, unless your installation has established that the account number and the programmer's name parameters must be coded. If no parameters are coded in the operand field of the JOB statement, no comments can be coded on the statement.

JOB Statement Format

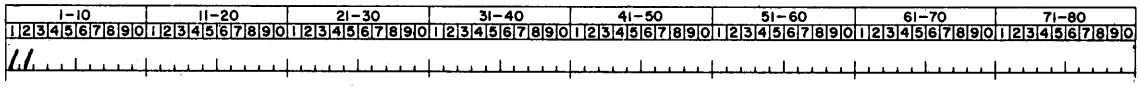
```
//jobname  JOB  operands  comments
```

The JOB statement consists of the characters //, in columns 1 and 2, and four fields -- the name, operation (JOB), operand, and comments fields.

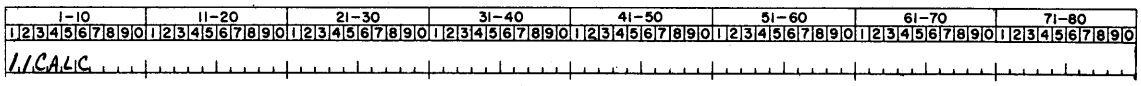
Rules for Coding

Follow the order listed below when coding the JOB statement:

1. Code the characters // in columns 1 and 2.

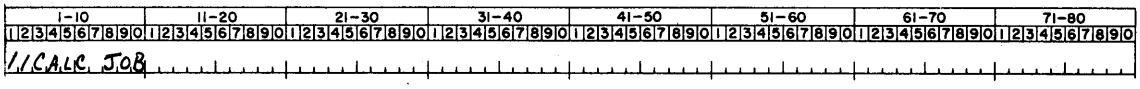


2. Select a name for the job; code that name, starting in column 3.



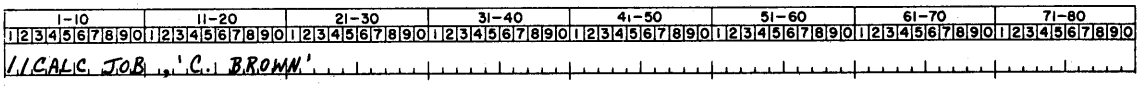
3. Follow the jobname with at least one blank.

4. Code JOB.

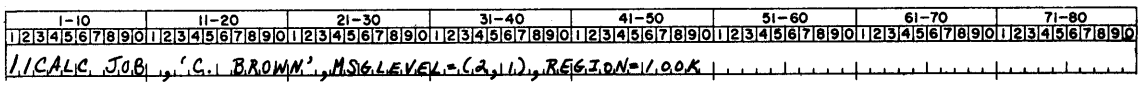


5. Follow JOB with at least one blank.

6. Code any desired positional parameters. Separate each parameter with a comma.

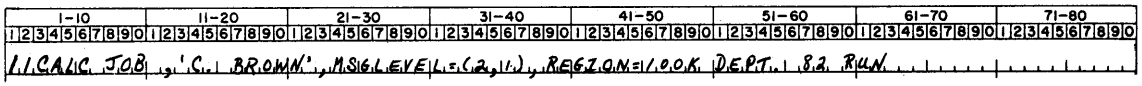


7. Code any desired keyword parameters. Separate each parameter with a comma.



8. Code at least one blank.

9. Code any desired comments.



Positional and Keyword Parameters

There are two types of parameters that can be coded on the JOB statement:

Positional parameters, which, when used, must precede any keyword parameters and must be coded in the following order:

accounting information
programmer's name

These positional parameters are described in the following pages in the order listed above.

Keyword parameters, which may be coded in any order after the positional parameters. Any of the following keyword parameters can be coded on the JOB statement:

CLASS
COND
MSGCLASS
MSGLEVEL
NOTIFY (MVT with TSO)
PRTY
RD
REGION (MVT only)
RESTART
ROLL (MVT only)
TIME
TYPRUN

These keyword parameters are described, after the positional parameters, in the order listed above.

Sample JOB Statements

```
1.  //ALPHA    JOB  843,LINEE,CLASS=F,MSGLEVEL=(1,1)
2.  //LOS      JOB  BROWNLY,REGION=90K,TIME=(4,30),MSGLEVEL=(2,0)
3.  //MART     JOB  1863,RESTART=STEP4
4.  //TRY8     JOB
```

Assigning a Jobname

```
//jobname  JOB
```

You must assign a name to every job submitted for execution. The jobname must begin in column 3 of the JOB statement and must consist of 1 through 8 alphameric and national (#, @, $) characters. The first character must be an alphabetic or national character.

No two jobs in a multiprogramming environment should have the same jobname.

The following names and characters should not be used as jobnames, because they are keywords of the DISPLAY command:

CONSOLES	A	U
DSNAME	N	
JOBNAMES	Q	
SPACE	R	
STATUS	T	

If you must assign one of these keywords as a jobname, notify the operator, so he will be sure to enclose the jobname in parentheses when he uses it with the DISPLAY command. For example, if you have assigned the jobname SPACE to a job and the system operator wishes to display the status of the job, he must issue a command stating DISPLAY (SPACE). If the parentheses were omitted, the operator would get the amount of available space on a particular direct access volume resulting from a DISPLAY SPACE command.

Examples of Valid Jobnames

```
//RERUN4      JOB

//#123A       JOB

//JOBD58      JOB
```

Accounting Information Parameter

```
([account number] [,additional accounting information,...])
```

account number
: the account number to which this job is to be charged.

additional accounting information
: any other accounting information required by an installation's accounting routines. When additional accounting information consists of more than one item, each must be separated by a comma.

Rules for Coding

1. When accounting information is supplied, it must be coded before any other parameter on the JOB statement.

2. The account number and each item of additional accounting information are considered subparameters and each must be separated by a comma.

3. When accounting information consists of more than one subparameter, you must enclose the information in either parentheses or apostrophes (5-8 punch), e.g., '5438,GROUP6' or (5438,GROUP6). If apostrophes are used, all accounting information enclosed in the apostrophes is considered as one field.

4. If the accounting information must be continued on another statement, enclose the accounting information in parentheses. You may not continue on another statement any accounting information enclosed in apostrophes.

5. The account number and other accounting information cannot exceed 142 characters, including the commas that separate the subparameters.

6. If any of the subparameters contain special characters (except hyphens), either: (1) enclose the accounting information in apostrophes, or (2) enclose the subparameter in apostrophes and the accounting information in parentheses, e.g., '5438,10/08/66' or (5438,'10/08/66'). (The enclosing apostrophes are not considered part of the information.) If one of the special characters is an apostrophe, code two consecutive apostrophes in its place, e.g., (5438,'O''NEILL'). If one of the special characters is an ampersand and you are not defining a symbolic parameter, code two consecutive ampersands in its place, e.g., '34&&8241'.

7. If you do not supply accounting information but do code the programmer's name, you must code a comma preceding the programmer's name to indicate that the accounting information parameter, which is a positional parameter, has been omitted. (Note: This may vary from one installation to another. Check whether you need to code a comma at your installation.)

Supplying Information Parameters

Accounting information is optional unless the installation establishes it as a requirement in a PARM field parameter of the cataloged procedure for the input reader.

Routines that process accounting information must be supplied by the installation. For information on how to add accounting facilities, refer to "Handling Accounting Information" in the Data Management for System Programmers publication.

Examples of the Accounting Information Parameter

1. `//JOB43 JOB D548-868`

   Account number only; no parentheses are required.

2. `//JOB44 JOB (D548-868,'12/8/69',WILSON)`

   Account number plus additional accounting information; parentheses are required.

3. `//JOB45 JOB (,F1659,GROUP12),GREGORY`

   Only additional accounting information; parentheses are required.

Programmer's Name Parameter

```
programmer's name
```

programmer's name
the name or identification of the person responsible for the job.

Rules for Coding

1. If the programmer's name parameter is coded, it must follow the accounting information parameter, or the comma that indicates its absence, and must precede all keyword parameters. (Note: This may vary from one installation to another. Check whether you need to code a comma at your installation.)

2. The name cannot exceed 20 characters, including all special characters.

3. If the name contains special characters, other than periods, enclose the name in apostrophes. If the special characters include apostrophes, each must be shown as two consecutive apostrophes.

4. If you are not required to specify a name, you need not code a comma to indicate its absence.

When to Code the Programmer's Name Parameter

The programmers' name parameter is optional unless the installation establishes it as a requirement in a PARM field parameter of the cataloged procedure for the input reader.

Examples of the Programmer's Name Parameter

1.
```
//APP      JOB  C.L.BROWN
```

   Programmer's name, without accounting information supplied.

2.
```
//DELTA    JOB  'T.O''NEILL'
```

   Programmer's name containing special characters, without accounting information supplied. (The leading comma is optional.)

3.
```
//#308     JOB  (846349,GROUP12),GREGORY
```

   Account number plus additional accounting information and programmer's name.

The CLASS Parameter

```
CLASS=jobclass
```

jobclass
assigns a job class to your job. Code any alphabetic character from A through O, depending on the characteristics of your job and the installation's rules for assigning a job class.

Rules for Coding

1. The jobclass is an alphabetic character from A through O.

Assigning a Job Class to Your Job

The CLASS keyword parameter provides a way of establishing a good mixture of jobs requiring different system resources. A good mixture can be established since the job class determines where a job will be placed on the input work queue and jobs with common characteristics are assigned to the same job class. Jobs within a job class are assigned a priority, either in the PRTY parameter or by default. This allows jobs within a class to be selected for processing based on their priorities.

If you do not specify the CLASS parameter, the default job class of A is assigned to the job.

THE CLASS PARAMETER AND TIME-SLICING

If your installation provides time-slicing facilities with MFT, the CLASS parameter can be used to make a job part of a group of jobs to be time-sliced. At system generation, a group of contiguous partitions are selected to be used for time-slicing, and each partition is assigned at least one job class. To make your job part of a group of jobs to be time-sliced, specify a class that was assigned only to the partitions selected for time-slicing. (With MVT, you use the PRTY parameter and the DPRTY parameter to make, respectively, a job or job step part of a group of jobs and job steps to be time-sliced.)

Examples of the CLASS Parameter

1. `//SETUP  JOB  CLASS=C`

   Assign a job to job class C.

2. `//JAN  JOB  CLASS=M,PRTY=10`

   Assigning a job to job class M with a priority of 10.

The COND Parameter

```
COND=((code,operator),...)
```


code
: a decimal number from 0 through 4095. This number is compared with the return code issued by each job step.

operator
: the type of comparison to be made with the return code. Relational operators and their meanings are:

  GT...greater than
  GE...greater than or equal to
  EQ...equal to
  LT...less than
  LE...less than or equal to
  NE...not equal to

Rules for Coding

1. Code from one through eight different return code tests.

2. When making only one return code test, you need not code the outer parentheses.

Using the COND Parameter

The COND keyword parameter can be used to eliminate unnecessary use of computing time by basing the continuation of a job on the completion of one or more of its job steps.

The operating system determines whether a job is to be discontinued after a given job step by comparing the return code produced by that job step to the conditions specified with the COND parameters. A return code is a number determined by the operating system or by the processing program which indicates the relative "success" of the job step. The return codes of the operating system and IBM-supplied processing programs are fixed numbers with specific meanings. They are listed in the publication IBM System/360 Operating System: Messages and Codes and in the publications associated with each processing program.

Only those user processing programs written in the assembler language, ANS COBOL, FORTRAN, or PL/I can set return codes for testing. The user return codes are usually standardized in each installation. For example, each step in your installation's payroll program may have its own set of return codes. One return code for a given job step may indicate that all payroll records were successfully processed while another may indicate that there were faulty input records. You can set up the COND parameter so that the job is discontinued if the return code that indicates faulty records is produced by that job step.

Not all return codes indicate either success or failure. For example, in the case of a compiler one return code can indicate no errors during compilation, a second code can indicate that the minor errors encountered are not likely to prevent link editing and execution of the compiled program, a third code can indicate that the major errors encountered will probably cause further processing of the compiled program to fail, and a fourth code can indicate that the compilation process has terminated abnormally. The COND parameter allows you to discontinue the job if any of these return codes are produced. You may

choose to continue processing only if no errors are found or, for debugging purposes, you may choose to continue processing even if major errors are found.

Note: If any job step is abnormally terminated (ABEND), all subsequent steps are bypassed unless the COND parameter of the EXEC statement is used to prevent it. (See the section on "The EXEC Statement.") If you want to restart the same step that terminated abnormally you can use the restart facilities of the operating system.

If you coded COND=((50,GE),(60,LT)), it would read "if 50 is greater than or equal to a return code, or 60 is less than a return code, I want the remaining job steps bypassed." In other words, the job continues as long as return codes range from 51 through 60. If you want to make only one return code test, you need not code the outer parentheses. For example, COND=(8,NE). A maximum of eight conditions can be established.

For example, if you code: COND=((5,GT),(8,EQ),(17,EQ),(19,EQ),(21,EQ),(23,LE)) your job will continue only if the return codes are: 5,6,7,9,10,11,12,13,14,15,16,18,20, or 22.

The tests you specify with the COND parameter are made to the return code, if any, produced by each step in your job. You can best take advantage of this parameter when the return codes of each job step have compatible meanings. For example, a return code of 4 from the ALGOL compiler indicates that the source program was compiled and some minor errors were found; the same return code of 4 from the linkage editor indicates that a load module was produced, but an error which may cause failure at execution time has been found. If you want to take a chance and continue processing even if small errors are found, you should code COND=(4,LT), that is, the job will terminate if the return code of any step is greater than 4. If you only want to continue processing if no errors are found, you should code COND=(4,LE), that is, the job will terminate if the return code of any step is greater than or equal to 4. (All codes greater than 4 indicate major errors for both the ALGOL compiler and the linkage editor.)

If the same return code has different meanings in different job steps, or if you want to take different actions according to which job step produced the return code, you should use the COND parameter of the EXEC statement to set up conditions for individual job steps.

If you omit the COND parameter from the JOB statement, no return code tests are performed throughout the job. If you want return codes tested for a given job step, use the COND parameter of the EXEC statement for that job step. If the COND parameter is not used in either the JOB or the EXEC statements, no return code tests are performed and the system will try to execute each step in the job.

If you code the COND parameter on the JOB statement and on one or more of the job's EXEC statements, the return code tests requested on the JOB statement have precedence over those requested on the EXEC statements. Therefore, any return code test requested on the JOB statement that is satisfied causes termination of the job, even if the return code test is not satisfied for a particular step.

Note: The COND parameter of the EXEC statement is slightly different from the COND parameter of the JOB statement. See the section on "The EXEC statement". Examples of using the COND parameter in both the JOB and EXEC statements are also shown in that section.

Examples of the COND Parameter

1. `//TYPE  JOB  COND=(7,LT)`

   If 7 is less than the return code, the job is terminated. (Any return code less than or equal to 7 allows the job to continue.)

2. `//TEST  JOB  COND=((20,GE),(30,LT))`

   IF 20 is greater than or equal to the return code, or 30 is less than the return code, the job is terminated. (Any return code of 21 through 30 allows the job to continue.)

The MSGCLASS Parameter

```
MSGCLASS=output class
```

output class
the output class to which system messages for your job are to be routed by the system. Code an alphabetic (A-Z) or numeric (0-9) character depending on your installation's rules for assigning an output class for system messages.

Rules for Coding

1. The output class is an alphabetic (A-Z) or numeric (0-9) character.

Assigning an Output Class to System Messages

If the MSGCLASS parameter is not coded, system messages associated with your job are routed to the default output class specified in the PARM field of the input reader procedure. The default for the MSGCLASS parameter is A unless changed by your installation. (Default values and restrictions supplied by IBM in the input reader procedure are listed in Appendix E. For more information on the input reader procedure, consult Data Management for System Programmers.) Your installation may require that you specify a different output class other than the default value in order to separate different types of output or to distribute the workload of the output writers. One or more output classes is associated with each output writer; each output writer is associated with a specific output device.

You can route a job's system messages and output data sets to the same output class. You do this by coding the same output class in both the MSGCLASS parameter on the JOB statement and the SYSOUT parameter on the DD statements for the data sets.

Examples of the MSGCLASS Parameter

1.
```
//IN      JOB  MSGCLASS=F
```

Specifying an output class.

2.
```
//BOILE   JOB
```

Specifying no output class. In this case, the output class will default to the MSGCLASS value specified in the PARM field of the input reader procedure. The default is A unless changed by your installation.

3.
```
//A1430   JOB  MSGCLASS=L
//STEP1   EXEC PGM=PRINT
//OUTPUT  DD   SYSOUT=L
```

Specifying that a job's system messages (MSGCLASS parameter) and output data set (SYSOUT parameter) are to be routed to the same output class.

The MSGLEVEL Parameter


```
MSGLEVEL=(statements,messages)
```

statements
specifies which job control statements are to be written as output from your job. Code:

0 - when only the JOB statement is to be written.

1 - when all input job control statements, cataloged procedure statements, and the internal representation of procedure statement parameters after symbolic parameter substitution are to be written.

2 - when only input job control statements are to be written.

messages
specifies what allocation/termination messages (consisting of allocation, disposition, and allocation recovery messages) are to be written as output from your job. Code:

0 - when no allocation/termination messages are to be written, unless the job abnormally terminates. If this occurs, these messages are to be written as output.

1 - when all allocation/termination messages are to be written.

Rules for Coding

1. If the first subparameter of the MSGLEVEL parameter is omitted, you must code a comma to indicate its absence, e.g., MSGLEVEL=(,1).

2. If the second subparameter of the MSGLEVEL parameter is omitted, you need not code the parentheses, e.g., MSGLEVEL=2.

Requesting Output of Job Control Statements and Certain Messages

The MSGLEVEL keyword parameter is used to tell the job scheduler what output from your job is to be written as part of the output listing. You can request the following output:

- The JOB statement.
- All input job control statements.
- All cataloged procedure statements for procedures called by any of the job's steps and the internal representation of procedure statement parameters after symbolic parameter substitution.
- Allocation, disposition, and allocation recovery messages (allocation/termination messages).

You need to code the MSGLEVEL parameter only when the established default will not provide you with the desired output. The default is established as a PARM parameter field in the cataloged procedure for the input reader. The established default is assumed when MSGLEVEL is not coded or when one of the subparameters is not coded. For system tasks, the system assumes a message level of (1,0).

Examples of the MSGLEVEL Parameter

1. `//GD40 JOB MSGLEVEL=(2,1)`

   Requesting that only input statements and all allocation/termination messages be written.

2. `//STEP JOB MSGLEVEL=(0,1)`

   Requesting that only the job statement and all allocation/termination messages be written.

3. `//SYM JOB MSGLEVEL=(1,0)`

   Requesting that all input control statements, procedure statements, the internal representation of procedure statements after symbloic parameter substitution, and no allocation/termination messages be written.

The NOTIFY Parameter (For MVT with TSO)

NOTIFY=user identification

user identification
specifies the identification that is to be used to notify you when your background job is complete. Code a 1 to 7 character alphameric identification. The first character must be an alphabetic character.

Rules for Coding

1. If the NOTIFY parameter is coded for MFT, or MVT without the Time Sharing Option (TSO), the parameter is not used, but is checked for syntax.

2. The user identification must be the same as the one you specify when you start the terminal session (LOGON).

What the NOTIFY Parameter Does

The NOTIFY keyword parameter indicates to the system that you are requesting that a message be sent to your time sharing terminal when your background job completes. Under TSO, a background job is one that is entered through the SUBMIT command or through the input stream (SYSIN).

WHAT IS TIME SHARING

Time sharing is a method of using a computing system that allows a number of users to execute programs concurrently and to interact with them during execution. The Time Sharing Option (TSO) is an option of the operating system providing conversational time sharing from remote terminals. That is the user "converses" with the system through the use of the terminal.

Reference:

1. For a detailed discussion of the Time Sharing Option, refer to IBM System/360 Operating System: Time Sharing Option Guide.

Example of the NOTIFY Parameter

1. `//SIGN  JOB  NOTIFY=POK1`

   When the job "SIGN" is complete, a message will be sent to the user "POK1" informing him that his job has been completed.

The PRTY Parameter

```
PRTY=priority
```

priority
assigns a priority of 0 through 13 to your job. (The highest priority is 13.)

Rules for Coding

1. Avoid using priority 13 since this priority is used by the system to expedite processing of jobs in which certain errors were diagnosed.

2. In MVT, if you want a job step to have a different dispatching priority than the job's, code the DPRTY parameter on the EXEC statement associated with that job step.

What the PRTY Parameter Does

The PRTY keyword parameter determines the job's initiation priority within its job class. (The job class is assigned in the CLASS parameter on the JOB statement.) When the job is initiated, the system converts the job's priority into a dispatching priority so that the job's tasks can compete with other tasks for use of main storage and CPU resources.

If you do not specify the PRTY parameter, a default priority is assumed. The default is specified as a PARM parameter field in the cataloged procedure for the input reader.

The PRTY Parameter and Time-Slicing

If your installation provides time-slicing facilities in MVT, the PRTY parameter can be used to make a job part of a group of jobs and job steps to be time-sliced. The priorities of the time-sliced groups are selected at system generation. To make your job part of a group of jobs to be time-sliced, specify a priority number selected for time-slicing. (To make one of the job's steps part of a group of jobs and job steps to be time-sliced, code the DPRTY parameter on the associated EXEC statement.)

Examples of the PRTY Parameter

1. `//#1930  JOB  PRTY=8,CLASS=C`

   The job will have an initiation priority of 8 in the job class C.

2. `//RING  JOB  PRTY=4`

   The job will have an initiation priority of 4 in the job class A. (Since the CLASS parameter is not specified, the job is assigned to the default job class A.)

The RD Parameter

```
RD=(R   )
   (RNC )
   (NC  )
   (NR  )
```


R
specifies that automatic step restart is permitted.

RNC
specifies that automatic step restart is permitted and automatic checkpoint restart is not permitted and no checkpoints can be established.

NC
specifies that neither automatic step restart nor automatic checkpoint restart is permitted and no checkpoints can be established.

NR
specifies that neither automatic step restart nor automatic checkpoint restart is permitted, but the CHKPT macro instruction can establish a checkpoint.

Rules for Coding

1. Be sure to code MSGLEVEL=(1,0), MSGLEVEL=(1,1), or MSGLEVEL=1 when RD=R or RD=RNC is specified.

2. If you are permitting automatic step restart, assign each step a unique step name.

3. Code the RD parameter on EXEC statements, instead of the JOB statement, when you want to make different restart requests for each job step. (If the RD parameter is coded on the JOB statement, RD parameters coded on the job's EXEC statements are ignored.)

Using the Restart Facilities

The RD (restart definition) keyword parameter is coded when you want to make use of the step restart facilities, to suppress the action of the CHKPT macro instruction, or to suppress automatic restarts. The step restart facilities permit execution of a job to be automatically restarted at a job step after the job abnormally terminates or after a system failure occurs. Through the RD parameter, you can specify that execution of a job is to be automatically restarted at the beginning of a job step that abnormally terminates (step restart).

Execution of a job can also be automatically restarted within a job step that abnormally terminates (checkpoint restart). In order for checkpoint restart to occur, the CHKPT macro instruction must have been executed in the processing program before abnormal termination. When you use the RD parameter to request suppression of CHKPT macro instruction action, automatic checkpoint restart cannot occur.

If the RD parameter is not coded, step restart cannot occur. If the RD parameter is not coded and the processing programs contain CHKPT macro instructions, checkpoint restart can occur.

The following three conditions must be met before automatic step or checkpoint restart can occur: (1) the completion code returned during abnormal termination indicates that the step is eligible for restart, (2) the operator authorizes restart, and (3) MSGLEVEL=(1,0), MSGLEVEL=(1,1), or MSGLEVEL=1 must be coded on the JOB statement. If these conditions are satisfied, special disposition processing is performed before restart. If automatic step restart is to occur, all data sets in the restart step with a status of OLD or MOD, and all data sets being passed to steps following the restart step, are kept. All data sets in the restart step with a status of NEW are deleted. If automatic checkpoint restart is to occur, all data sets currently in use by the job are kept.

DEFINING RESTART

You define the type of restart that can occur by coding one of the subparameters of the RD parameter: R, RNC, NC, or NR. Each of these subparameters is described in detail in the following paragraphs.

RD=R: R indicates that automatic step restart is permitted. If the job's processing programs do not include any CHKPT macro instructions, coding RD=R permits execution to be resumed at the beginning of any step that abnormally terminates. If any program does include a CHKPT macro instruction, coding RD=R permits step restart to occur only if the step abnormally terminates before execution of the CHKPT macro instruction; thereafter, only checkpoint restart can occur. If you cancel the effects of the CHKPT macro instruction before a checkpoint restart is performed, the request for automatic step restart is again in effect.

RD=RNC: RNC indicates that automatic step restart is permitted and automatic checkpoint restart is not permitted. RD=RNC should be specified when you want to suppress the action of all CHKPT macro instructions included in the job's processing programs and to permit automatic step restart.

RD=NC: NC indicates that neither automatic step restart nor automatic checkpoint restart is permitted. RD=NC should be specified when you want to suppress the action of all CHKPT macro instructions included in the job's processing programs and not to permit automatic step restart. RD=NC has no effect on processing if CHKPT macro instructions are not included in the programs.

RD=NR: NR indicates that a CHKPT macro instruction can establish a checkpoint, but neither automatic step restart nor automatic checkpoint restart is permitted. Coding RD=NR allows you to resubmit the job at a later time and specify in the RESTART parameter the checkpoint at which execution is to be resumed. (The RESTART parameter is coded on the JOB statement of the resubmitted job.) RD=NR has no effect on processing if CHKPT macro instructions are not included in the job's processing programs.

References

1. For detailed information on the checkpoint/restart facilities, refer to the publication Advanced Checkpoint/Restart Planning Guide, Form C28-6708, the topic "Checkpoint and Restart" in the publication Supervisor Services, and "Using the Restart Facilities" in Appendix B of this publication.

2. For information on how to code the CHKPT macro instruction, refer to the publication Supervisor Service and Macro Instructions.

Examples of the RD Parameter

1. `//MAY  JOB  RD=R,MSGLEVEL=(1,0)`

   Permits execution to be automatically restarted with the step that abnormally terminates.

2. `//TRY56  JOB  RD=RNC,MSGLEVEL=1`

   Permits execution to be automatically restarted beginning with the step that abnormally terminates and suppresses the action of CHKPT macro instructions.

3. `//PASS  JOB  RD=NR,MSGLEVEL=(1,1)`

   Neither automatic step nor checkpoint restart can occur, but CHKPT macro instructions can establish checkpoints.

The REGION Parameter - Without Main Storage Hierarchy Support (For MVT)

```
REGION=valueK
```

valueK
"value" specifies the number of contiguous 1024-byte areas of main storage to be allocated to each job step. The number can range from one to five digits but may not exceed 16383.

Rules for Coding

1. Code an even number followed by a "K". (If you code an odd number, the system treats it as the next highest even number. When the value 16383K is coded, the system treats it as 16384K. However, the value 16384K must not be coded on the JOB statement.)

2. Code the REGION parameter on EXEC statements, instead of the JOB statement, when you want to specify a different region size for each job step. (If the REGION parameter is coded on the JOB statement, REGION parameters coded on the job's EXEC statements are ignored.)

3. If the REGION parameter is coded for MFT, the parameter is not used, but is checked for syntax.

Requesting Main Storage

The REGION keyword parameter is used to specify how much main storage, in contiguous bytes, is to be allocated to each job step. Code the region parameter when you want more storage or less storage than would be allocated if the default region size was used; the default value is used if you do not code the REGION parameter on either the JOB or EXEC statement. The default region size is established as a PARM parameter field in the cataloged procedure for the input reader. You can consult the Storage Estimates publication to help you determine how much main storage is required to process your job.

AQUIRING ADDITIONAL MAIN STORAGE

If any of the job's steps may require use of more storage than has been allocated, you can code the ROLL parameter and request that the system try to provide you with additional main storage. The ROLL parameter is described in the chapters "The ROLL Parameter" later in this section and in Section III.

Examples of the REGION Parameter

1.
```
//COLE     JOB  REGION=112K
```

   Specifies that 112 contiguous 1024-byte areas of main storage are to be allocated to each job step.

2.
```
//J34      JOB  REGION=70K,ROLL=(YES,YES)
```

   The REGION parameter specifies that 70 contiguous 1024-byte areas of main storage are to be allocated to each job step. In the ROLL parameter, the first subparameter tells the system that any of the job's steps may be rolled out if additional storage is required by another job; the second subparameter tells the system that it should try to provide you with main storage if it is required.

The REGION Parameter - With Main Storage Hierarchy Support (For MVT, Excluding M65MP)

REGION=(valueK,value$_1$K)

valueK
specifies the number of contiguous 1024-byte areas in hierarchy 0 to be allocated to each job step. If IBM 2361 Core Storage is present, the number cannot exceed 16383.

value$_1$K
specifies the number of contiguous 1024-byte areas in hierarchy 1 to be allocated to each job step. If IBM 2361 Core Storage is present, the number cannot exceed 1024 (for each Model 1) or 2048 (for each Model 2).

Rules for Coding

1. When processor storage includes hierarchies 0 and 1, the sum of value and value$_1$ cannot exceed 16383.

2. Code even numbers. (If you code an odd number, the system treats it as the next highest even number. When 16383K is coded, the system treats it as 16384K. However, a sum of 16384K must not be coded on the JOB statement.)

3. When you are requesting storage only in hierarchy 1, precede value$_1$ with a comma, to indicate the absence of "value".

4. When you are requesting storage only in hierarchy 0, you need not code the parentheses.

5. Code the REGION parameter on EXEC statements, instead of the JOB statement, when you want to specify a different region size for each job step. (If the REGION parameter is coded on the JOB statement, REGION parameters coded on the job's EXEC statements are ignored.)

6. If the REGION parameter is coded for MFT, the parameter is not used, but is checked for syntax.

Requesting Main Storage in One or Two Hierarchies

The REGION keyword parameter is used to specify how much main storage is to be allocated to each job step, and, when main storage hierarchy support has been specified at system generation, in which hierarchy or hierarchies main storage is to be allocated. With main storage hierarchy support, storage hierarchies 0 and 1 are provided. If IBM 2361 Core Storage, Model 1 or 2, is present in the system, processor storage is referred to as hierarchy 0 and 2361 Core Storage is referred to as hierarchy 1. If 2361 Core Storage is not present, a two-part region is established in processor storage when regions are requested in two hierarchies. The two parts are not necessarily contiguous.

Code the REGION parameter to specify how much storage is to be allocated in each hierarchy, or that all storage for the job is to be allocated in a particular hierarchy. (If you do not code the REGION parameter on either the JOB or EXEC statement, the default region size, which is a PARM parameter field in the cataloged procedure for the input reader, is used and is always allocated in hierarchy 0. If you code the REGION parameter and request storage only from hierarchy 1, no hierarchy 0 segment will be allocated. You can consult the Storage Estimates publication to help you determine how much main storage is required to

process your job. Then, depending on your reasons for using hierarchies, determine how much storage is required in each.

If main storage hierarchy support was not specified at system generation and regions are requested in both hierarchies, the region sizes are combined and an attempt is made to allocate a single region from processor storage. If a region is requested entirely from hierarchy 1, an attempt is made to allocate the region from processor storage.

ACQUIRING ADDITIONAL MAIN STORAGE

If your job may require use of more main storage than has been allocated in a particular hierarchy, you can code the ROLL parameter and request that the system try to provide you with additional main storage in that hierarchy. The ROLL parameter is described in the chapters "The ROLL Parameter" later in this section and in Section III.

Examples of the REGION Parameter

1. `//MAIN  JOB  REGION=(80K,30K)`

   Specifies that the system is to allocate 80 contiguous 1024-byte areas of storage in hierarchy 0 and 30 contiguous 1024-byte areas of storage in hierarchy 1. If main storage hierarchy support is not included in the system, the system will try to obtain 110 contiguous 1024-byte areas in processor storage.

2. `//WEEK  JOB  REGION=(,98K)`

   Specifies that the system is to allocate 98 contiguous 1024-byte areas of storage in hierarchy 1.

3. `//JWC  JOB  REGION=98K`

   Specifies that the system is to allocate 98 contiguous 1024-byte areas of storage in hierarchy 0.

4. `//TEST12  JOB  REGION=(100K,50K),ROLL=(YES,YES)`

   The REGION parameter specifies that the system is to allocate 100 contiguous 1024-byte areas of storage in hierarchy 0 to 50 contiguous 1024-byte areas of storage in hierarchy 1. In the ROLL parameter, the first subparameter tells the system that any of the job's steps may be rolled out if additional storage is required by another job; the second subparameter tells the system that it should try to provide you with additional main storage if it is required.

The RESTART Parameter

```
RESTART=( { *                        } [,checkid])
          { stepname                 }
          { stepname.procstepname    }
```


*
indicates that execution is to be restarted at or within the first job step.

stepname
specifies that execution is to be restarted at or within the named job step.

stepname.procstepname
specifies that execution is to be restarted at or within a cataloged procedure step. Stepname is the name of the job step that calls the cataloged procedure, and procstepname is the name of the procedure step. You can code * in place of stepname.procstepname if the first job step calls a cataloged procedure and you want execution to be restarted at or within the first procedure step.

checkid
is the name of the checkpoint at which execution is to be restarted. When checkid is coded, execution is restarted within the specified job step at the named checkpoint. If checkid is not coded, execution is restarted at the specified job step.

Rules for Coding

1. You need not code the parentheses if execution is to be restarted at a job step, i.e., if you do not code the checkid subparameter.

2. If the checkpoint name contains special characters, the name must be enclosed in apostrophes. If one of the special characters is an apostrophe, identify it by coding two consecutive apostrophes in its place.

3. Be sure to include the SYSCHK DD statement when execution is to be restarted within a job step. (The SYSCHK DD statement is described in the section titled "SYSCHK" in the chapter "Assigning a Ddname" in Section IV of this publication.)

When to Code the RESTART Parameter

The RESTART keyword parameter is coded when you are resubmitting a job for execution and you want to make use of the restart facilities. The restart facilities allow a job that is resubmitted for execution to be restarted at or within a particular job step. This reduces the time required to execute the job since execution is resumed, not repeated. If the RESTART parameter is not coded, execution of the entire job is repeated.

Through the RESTART parameter, you can specify where execution is to be restarted. Execution of a resubmitted job can be restarted at the beginning of a step (step restart) or within a step (checkpoint restart). In order for checkpoint restart to occur, the CHKPT macro instruction must have been executed in the processing program during the original execution of the job. If execution is to be restarted at a checkpoint, the resubmitted job must include an additional DD statement. This DD statement defines the checkpoint data set and has the ddname SYSCHK. (For additional information on the SYSCHK DD statement, see the

section titled "SYSCHK" in the chapter "Assigning a Ddname" in Section IV of this publication.)

RULES FOR REFERENCING GENERATION DATA SETS AND USING BACKWARD REFERENCES

Because the resubmitted job has been previously executed and because you may not be restarting with the first job step, there are certain rules that apply to referencing generation data sets and using backward references. They are:

1. If step restart is performed, generation data sets that were created and cataloged in steps preceding the restart step must not be referred to in the restart step or in steps following the restart step by means of the same relative generation numbers that were used to create them. Instead, you must refer to a generation data set by means of its present relative generation number. For example, if the last generation data set created and cataloged was assigned a generation number of +2, it would be referred to as 0 in the restart step and in steps following the restart step. In this case, the generation data set assigned a generation number of +1 would be referred to as -1. If generation data sets created in the restart step were kept instead of cataloged (i.e., DISP=(NEW,CATLG,KEEP) was coded), you can during checkpoint restart refer to these data sets and generation data sets created and cataloged in steps preceding the restart step by the same relative generation numbers used to create them.

2. Before resubmitting a job, check all backward references to steps that precede the restart step. Eliminate all backward references for the following keywords: PGM and COND, on the EXEC statements, and, SUBALLOC and VOLUME=REF=reference, on the DD statements. (A backward reference of VOLUME=REF=reference is allowed if the referenced statement includes VOLUME=SER=(serial number,...).)

Reference

1. For detailed information on the checkpoint/restart facilities, refer to the publication Advanced Checkpoint/Restart Planning Guide, the topic "Checkpoint and Restart" in the publication Supervisor Services and "Using the Restart Facilities" in Appendix B of this publication.

Examples of the RESTART Parameter

1. `//LINES  JOB  RESTART=COUNT`

   Specifies that execution is to be restarted at the job step named COUNT.

2. `//@LOC5  JOB  RESTART=(PROCESS,CHKPT3)`

   Specifies that execution is to be restarted within the job step named PROCESS at the checkpoint named CHKPT3. This JOB statement must be followed by a DD statement named SYSCHK, which defines the data set or which an entry for the checkpoint named CHKPT3 was written.

3. `//WORK  JOB  RESTART=(*,CKPT2)`

   Specifies that execution is to be restarted at the checkpoint named CKPT2 in the first job step.

4. `//CLIP5  JOB  RESTART=(PAY.WEEKLY,CHECK8)`

   Specifies that execution is to be restarted within the procedure step named WEEKLY at the checkpoint named CHECK8. PAY is the name of the job step that calls the cataloged procedure that contains the procedure step named WEEKLY. This JOB statement must be followed by a DD statement named SYSCHK, which defines the data set on which an entry for the checkpoint named CHECK8 was written.

The ROLL Parameter (For MVT)

ROLL=(x,y)

x
declares whether the steps of the job may be rolled out. Code YES if the job's steps can be rolled out; code NO if the job's steps cannot be rolled out.

y
declares whether the steps of the job may cause rollout of another job step. Code YES if the job's steps can cause rollout of another job step; code NO if the job's steps cannot cause rollout of another job step. YES must be coded if you want additional main storage allocated to the job's steps when additional main storage is required.

Rules for Coding

1. If you code the ROLL parameter, both subparameters must be specified.

2. Code the ROLL parameter on EXEC statements, instead of the JOB statement, when you want to make different requests for each job step. (If the ROLL parameter is coded on the JOB statement, ROLL parameters coded on the job's EXEC statements are ignored.)

3. Code ROLL=(NO,YES) or ROLL=(NO,NO) if this job is a teleprocessing job that uses the Auto Poll option. If you allow the job's steps to be rolled out, the job cannot be restarted properly.

4. If the ROLL parameter is coded for MFT, the parameter is not used, but is checked for syntax.

When to Code the ROLL Parameter

The ROLL keyword parameter should be coded if any of the job's steps may require more main storage than was requested in the REGION parameter. When you specify in the ROLL parameter that this job can cause rollout of other job steps, an attempt is made to allocate additional storage if a job step requires it. In order to allocate this additional space to a job step, another job step with a lower priority may have to be rolled out, i.e., temporarily transferred to secondary storage.

The ROLL parameter should also be coded when you want control over whether the job's steps can be rolled out because of another step's need for additional main storage. If the ROLL parameter is not coded, the default established in the PARM parameter field in the cataloged procedure for the input reader is used.

Examples of the ROLL Parameter

1. `//DINTER JOB ROLL=(YES,YES),REGION=100K`

   Specifies that the job's steps can be rolled out and can cause rollout of another job step if a step requires more than 100K of main storage.

2. `//TEST332 JOB ROLL=(NO,YES)`

   Specifies that the job's steps cannot be rolled out but can cause rollout of another job step.

The TIME Parameter

```
TIME=  {(minutes,seconds)}
       {1440             }
```


minutes
specifies the maximum number of minutes the job can use the CPU. The number of minutes must be less than 1440 (24 hours).

seconds
specifies the maximum number of seconds beyond the specified number of minutes the job can use the CPU, or, if no minutes are specified, the maximum number of seconds the job can use the CPU. The number of seconds must be less than 60.

1440
specifies that the job is not to be timed. Code 1440 if the job may require use of the CPU for 24 hours or more or if any of the job's steps should be allowed to remain in a wait state for more than the established time limit.

Rules for Coding

1. If the CPU time limit is given in minutes only, you need not code the parentheses.

2. If the CPU time limit is given in seconds only, you must code a comma preceding the seconds to indicate the absence of minutes.

3. You can also code the TIME parameter on EXEC statements to indicate how long each step can use the CPU.

Specifying a Time Limit for the Job

The TIME keyword parameter can be used to specify the maximum amount of time a job may use the CPU. (CPU time will appear on the output listing if system management facilities (SMF) or users accounting routines supply this information to the output data set.) By coding the TIME parameter, you can limit the CPU time wasted by a step that goes into a loop. Normally, a job that exceeds the specified time limit is terminated. However, if the System Management Facilities option is included in the system and a user exit routine is provided, this routine can extend the time limit so that processing can continue. When the TIME parameter is not coded on the JOB statement, there is no CPU time limit assigned to the job; however, each job step is still timed.

TIME LIMIT FOR WAIT STATES

Since a job step can go into an extremely long wait state, the time a job step may remain in a wait state is limited. If the System Management Facilities option is included in the system, the installation determines this time limit. In this case, a job step remaining in a wait state for more than the established time limit causes termination of the job unless a user-provided exit routine extends the wait-state time limit for that step. If the System Management Facilities option is not included, the system automatically provides a 30-minute time limit for wait states; a job step remaining in a wait state for more than 30 consecutive minutes causes termination of the job.

How to Eliminate Timing

Certain applications require that a job use the CPU for 24 hours or more. In these cases you must eliminate job and step timing by coding TIME=1440. You should also code TIME=1440 when any of the job's steps should be allowed to remain in a wait state for more than the established time limit.

If your system includes the System Management Facilities (SMF) feature and you code TIME=1440, SMF termination messages will indicate that no CPU time was used; messages indicating the time the job started and stopped will not, however, be affected.

Reference

1. A discussion of the System Management Facilities option is contained in the Introduction. Information on user exit routines to be used with the System Management Facilities option is contained in the System Management Facilities Guide.

Examples of the TIME Parameter

1. `//SEED JOB TIME=(12,10)`

   Specifies that the maximum amount of time the job can use the CPU is 12 minutes 10 seconds.

2. `//TYPE41 JOB TIME=(,30)`

   Specifies that the maximum amount of time the job can use the CPU is 30 seconds.

3. `//FORMS JOB TIME=5`

   Specifies that the maximum amount of time the job can use the CPU is 5 minutes.

4. `//RAINCK JOB TIME=1440`

   Specifies that the job is not to be timed. Therefore, the job may use the CPU and may remain in a wait state for an unspecified period of time.

The TYPRUN Parameter (For MFT, MVT)

```
TYPRUN=HOLD
```

HOLD
specifies that the job is to be held in the job queue until the operator issues a RELEASE command.

Holding a Job

Code TYPRUN=HOLD when the job should be held for execution until some event has occurred. The operator must be informed of what it is you are waiting for. When the event has occurred, the operator issues a RELEASE command, thereby allowing the job to be selected for processing.

Example of the TYPRUN Parameter

Jobs UPDATE and LIST are to be submitted for execution. The job UPDATE uses a program that adds and deletes members of a library; the job LIST uses a program that lists the members of a library. In order to get an up-to-date listing of the library, UPDATE must be executed before LIST. This is accomplished by coding TYPRUN=HOLD on the JOB statement for the job named LIST. If a DISPLAY JOBNAMES command is issued by you or the operator, the operator is notified on the console when UPDATE has completed processing; he issues a RELEASE command for LIST. The job LIST can then be selected for execution.

Section III: The EXEC Statement

The EXEC statement is the first statement of each job step and cataloged procedure step. The EXEC statement is followed by DD statements and data that pertain to the step. The principal function of the EXEC statement is to identify the program to be executed or the cataloged procedure to be called. All other parameters in the operand field are optional. A job cannot contain more than 255 job steps and procedure steps.

EXEC Statement Format

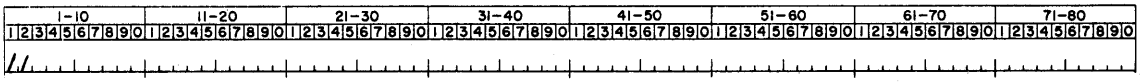
```
//stepname  EXEC  operands  comments
```

The EXEC statement consists of the characters //, in columns 1 and 2, and four fields -- the name, operation (EXEC), operand, and comments fields.

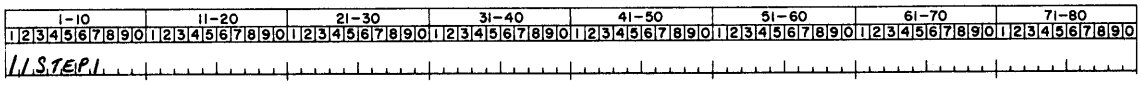
Rules for Coding

Follow the order listed below when coding the EXEC statement:

1. Code the characters // in columns 1 and 2.

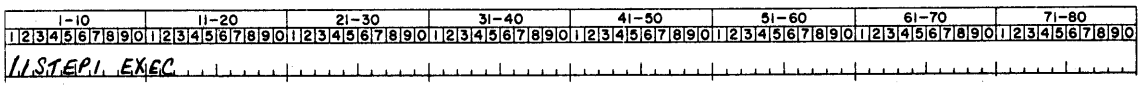


2. Optionally, you may assign a name to the job step; if you do, code the stepname starting in column 3.



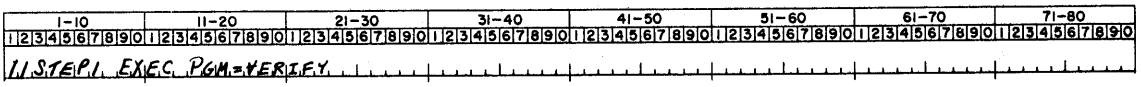
3. Follow the stepname or // with at least one blank.

4. Code EXEC.

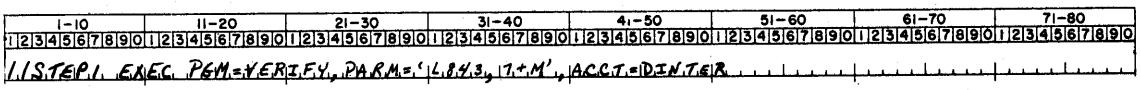


5. Follow EXEC with at least one blank.

6. Identify the program to be executed (PGM), or the cataloged procedure to be called (PROC). (When you are calling a procedure, you may omit PROC=.)

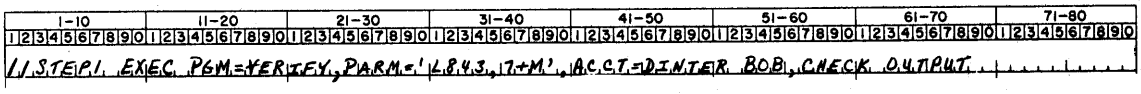


7. Code any desired keyword parameters. Separate each parameter with a comma.



8. Code at least one blank.

9. Code any desired comments.



Positional and Keyword Parameters

There are two types of parameters that can be coded on the EXEC statement:

Positional parameters must precede any keyword parameters. One of the following two parameters is coded:

PGM
PROC

These positional parameters are described in the following pages.

Keyword parameters may be coded in any order after the first parameter. Any of the following keyword parameters can be coded on the EXEC statement:

ACCT
COND
DPRTY (MVT only)
PARM
RD
REGION (MVT only)
ROLL (MVT only)
TIME

These keyword parameters are described, after the positional parameters, alphabetically.

Sample EXEC Statements

1. `//STEP4  EXEC PGM=DRBC,PARM='3018,NO'`

2. `//  EXEC PGM=ENTRY,REGION=80K,TIME=(2,30),DPRTY=(11,11)`

3. `//FOR  EXEC PROC=PE489,TIME=4`

4. `//PIC4  EXEC SAL83,ACCT.STEP1=123019`

Assigning a Stepname

```
//stepname  EXEC
```

The stepname identifies a job step within a job. The stepname is optional. You must assign a stepname if you wish to do any of the following:

1. Make backward references to the step.
2. Override parameters on an EXEC statement or DD statement in a cataloged procedure step, and add DD statements to a cataloged procedure step.
3. Perform a step or checkpoint restart at or within the step.

The stepname must begin in column 3 of the EXEC statement and must consist of 1 through 8 alphameric and national (@, #, $) characters. The first character must be an alphabetic or national character. Each stepname within a job or a cataloged procedure must be unique.

Examples of Valid Stepnames

1. `//STEP4  EXEC`
2. `//@LOC  EXEC`
3. `//PRINT  EXEC`

The PGM Parameter

```
PGM={program name                        }
    {*.stepname.ddname                   }
    {*.stepname.procstepname.ddname      }
```

program name
is the member name or alias of the program to be executed. The program must be a member of a partitioned data set that resides in a temporary, system, or private library.

*.stepname.ddname
is a backward reference to a DD statement that defines, as a member of a partitioned data set, the program to be executed; stepname is the name of the step in which the DD statement appears. Usually, this form is used when a previous job step creates a temporary partitioned data set to store one program until the program is required.

*.stepname.procstepname.ddname
is a backward reference to a DD statement within a cataloged procedure step that defines, as a member of a partitioned data set, the program to be executed. Stepname is the name of the step that calls the procedure, and procstepname is the name of the procedure step that contains the DD statement. Usually, this form is used when a cataloged procedure step, called by an earlier job step in the job, creates a temporary partitioned data set to store a program until the program is required.

Note: The stepname must begin in column 3 of the EXEC statement and must consist of 1 through 8 alphameric or national (@, #, $) characters. The first character must be an alphabetic or national character. Each stepname within a job or a cataloged procedure must be unique.

Identifying the Program to be Executed

All programs that can be executed are members of partitioned data sets (libraries). The library that contains the program may be a temporary library, the system library, or a private library. In order to execute a program contained in any of these libraries, you must code the PGM parameter as the first parameter on the EXEC statement.

TEMPORARY LIBRARY

If in a job you want to assemble, linkage edit, and then execute a program, you must make the output of the linkage editor a member of a partitioned data set. This is accomplished by creating a temporary library. A temporary library is a partitioned data set created in the job to store a program, as a member of the data set, until it is executed in a following job step. When the program is required, you may refer back to the DD statement that defines the temporary library and the member by coding PGM=*.stepname.ddname or PGM=*.stepname.procstepname.ddname. You may also request use of a program that is a member of a temporary library by coding PGM=program name and including a DD statement named JOBLIB or STEPLIB that defines the temporary library. (Information on the JOBLIB and STEPLIB DD statements can be found in the chapter "Assigning a Ddname" in Section IV of this publication.)

If you want to keep this program available for use by other jobs, you must make the program a member of the system library or a private library.

SYSTEM LIBRARY

The system library is a partitioned data set named SYS1.LINKLIB and it contains frequently used programs, as well as programs used by the system. You request the use of a program that is a member of the system library simply by coding PGM=program name. The system automatically looks in SYS1.LINKLIB for a member with the corresponding name.

A program that resides in the system library may also be executed by coding PGM=*.stepname.ddname or PGM=*.stepname.procstepname.ddname. This can be done only when the named DD statement defines the program as a member of the system library.

PRIVATE LIBRARY

A private library is a partitioned data set that contains programs not used frequently enough to warrant their inclusion in the system library. You request use of a program that is a member of a private library by coding PGM=program name and including a DD statement named JOBLIB or STEPLIB that defines the private library. The system automatically looks in the private library and, if the program is not found there, in SYS1.LINKLIB for a member with the corresponding name. (Information on the JOBLIB and STEPLIB DD statements can be found in the sections titled "JOBLIB" and "STEPLIB" in the chapter "Assigning a Ddname" in Section IV of this publication.)

A program that resides in the private library may also be executed by coding PGM=*.stepname.ddname or PGM=*.stepname.procstepname.ddname. This can be done only when the named DD statement defines the program as a member of a private library.

THE IEFBR14 PROGRAM

This is a small program which, when called, gives a return code of 0 and returns to the calling routine. Its purpose is for either checking the syntax of the control statements or for allocating or unallocating data sets prior to executing your program. To use this program, substitute IEFBR14 for your program's name. (If you created a data set when using this program, the data set's status will be old when you execute your own program.)

Examples of the PGM Parameter

1.
```
//STEP1   EXEC PGM=TABULATE
```

Specifies that the program named TABULATE is a member of SYS1.LINKLIB.

2.
```
//JOB8    JOB  MSGLEVEL=(2,0)
//JOBLIB  DD   DSNAME=DEPT12.LIB4,DISP=(OLD,PASS)
//STEP1   EXEC PGM=USCAN
```

Specifies that the system is to look for a program named USCAN in a private library named DEPT12.LIB4, and, if not found there, the system is to look in the system library.

3.
```
//STEP2   EXEC PGM=UPDT
//DDA     DD   DSNAME=SYS1.LINKLIB(P40),DISP=OLD
//STEP3   EXEC PGM=*.STEP2.DDA
```

Use of backward reference to a DD statement that defines the system library. The program named P40 is stored as a member of SYS1.LINKLIB and is executed in the step named STEP3.

4.
```
//CHECK   EXEC PGM=IEFBR14
```

Executing the program named IEFBR14 allows you to satisfy space allocation and disposition processing requests prior to executing your program. The remaining job control statements in the job are also checked for syntax.

The PROC Parameter

```
{PROC=procedure name}
{procedure name     }
```

procedure name
the member name (or alias) of the cataloged procedure or the name of the in-stream procedure to be called.

Identifying the Cataloged or In-Stream Procedure to be Called

A cataloged procedure is a set of job control statements that has been placed in a special partitioned data set referred to as the procedure library. (The IBM-supplied procedure library is named SYS1.PROCLIB; at particular installations, there may be additional procedure libraries, which would have different names.) Each cataloged procedure is a member of this data set. An in-stream procedure is a set of job control statements, beginning with a PROC statement and ending with a PEND statement, that have been placed in the input stream. An in-stream procedure can be executed any number of times during the job in which it appears. Both cataloged and in-stream procedures consist of one or more procedure steps; each procedure step consists of an EXEC statement, which identifies the program to be executed, and DD statements, which define the data set requirements of the step.

In order to use a cataloged or in-stream procedure, you must code the PROC statement as the first parameter on the EXEC statement, instead of the PGM parameter, and give the name of the cataloged procedure. You can, instead, code only the cataloged or in-stream procedure name; the job scheduler will recognize that it is a procedure name since it must appear first in the operand field.

When the EXEC statement specifies that a cataloged or in-stream procedure is to be called, subsequent parameters in the operand field can be used to override EXEC statement parameters in the procedure. Also, any DD statements that follow the EXEC statement are either overriding DD statement or DD statements that are to be added to the cataloged or in-stream procedure for the duration of the job step. Overriding or adding to cataloged procedures is discussed in the chapter "Using Cataloged and In-stream Procedures" in Appendix A of this publication.

Examples of the PROC Parameter

1. `//SP3  EXEC PROC=PAYWKRS`

   Specifies that the cataloged or in-stream procedure named PAYWKRS is to be called.

2. `//BK3  EXEC OPERATE`

   Specifies that the cataloged or in-stream procedure named OPERATE is to be called. This specification has the same effect as coding PROC=OPERATE.

The ACCT Parameter

```
ACCT=(accounting information,...)
```

accounting information
includes one or more subparameters of accounting information to be passed to the installation's accounting routines by the system.

Rules for Coding

1. If the accounting information includes several subparameters, each must be separated by a comma.

2. If the accounting information consists of only one subparameter, you need not code the parentheses.

3. The maximum number of characters of accounting information, plus the commas that separate the subparameters, is 142.

4. If a subparameter contains special characters (other than a hyphen), enclose the subparameter in apostrophes. The apostrophes are not considered part of the information. If one of the special characters is an apostrophe, code two consecutive apostrophes in its place. The same is true for the special character &. In order to include & within the apostrophes, code &&.

Providing Accounting Information for a Job Step or Procedure Step

Code the ACCT keyword parameter when you want to provide accounting information for a step. If the job step calls a cataloged procedure, the ACCT parameter overrides any ACCT parameters coded in the procedure steps and pertains to all the procedure steps. If different steps in the procedure require different accounting information, code ACCT.procstepname=(accounting information,...) for each step that requires accounting information. Accounting information will then pertain only to the named procedure step.

Examples of the ACCT Parameter

1.
```
//STEP1   EXEC PGM=JP5,ACCT=(LOCATION8,'CHGE+3')
```

Specifies that this accounting information pertains to this job step.

2.
```
//STP3    EXEC LOOKUP,ACCT=('/83468')
```

Specifies that this information pertains to this job step. Since this step calls a cataloged procedure, the accounting information pertains to all the steps in the procedure.

3.
```
//STP4    EXEC BILLING,ACCT.PAID=56370,ACCT.LATE=56470,
//            ACCT.BILL='121+366'
```

Specifies that different accounting information pertains to each of the named procedure steps (PAID, LATE, and BILL).

The COND Parameter

```
COND=(  [(code,operator)                          ]  ,... [,EVEN]  )
        [(code,operator,stepname)                 ]       [,ONLY]
        [(code,operator,stepname.procstepname)    ]
```

code
a decimal number from 0 through 4095. This number is compared with the return code issued by all previous steps or a specific step.

operator
the type of comparison to be made with the return code. Relational operators and their meanings are:

GT...greater than
GE...greater than or equal to
EQ...equal to
LT...less than
LE...less than or equal to
NE...not equal to

stepname
the name of a preceding job step that issued the return code to be tested.

stepname.procstepname
the name of a procedure step "procstepname" that issued the return code to be tested; the procedure step is part of a procedure that was called by an earlier job step named "stepname."

EVEN
specifies that the job step is to be executed even if one or more of the preceding job steps have abnormally terminated. If the current job step specifies that return code tests are to be made and if any of the tests are satisfied, this job step is bypassed. Do not code EVEN when ONLY is coded.

ONLY
specifies that the job step is to be executed only if one or more of the preceding job steps have abnormally terminated. If the current job step specifies that return code tests are to be made and if any of the tests are satisfied, this job step is bypassed. Do not code ONLY when EVEN is coded.

Rules for Coding

1. When neither EVEN nor ONLY is coded, you can make as many as eight tests on return codes issued by preceding job steps or cataloged procedure steps, which completed normally. When either EVEN or ONLY is coded, you can make as many as seven tests on return codes.

2. If you want only one test made, you need not code the outer parentheses.

3. If you code only EVEN or ONLY, you need not enclose it in parentheses.

4. If you want each return code test to be made on the return code issued by every preceding step, do not code a stepname.

5. The EVEN or ONLY subparameter can appear before, between, or after return code tests.

Using the COND Parameter

The COND keyword parameter can be used to eliminate unnecessary use of computing time by basing the execution of a job step on the successful completion of one or more preceding job steps. When the COND parameter is coded on the JOB statement, any return code test that is satisfied causes all remaining job steps to be bypassed. If, instead, you want a particular job step to be bypassed when a return code test is satisfied, code the COND parameter on the EXEC statement. Besides allowing you to specify the conditions for bypassing a job step, the COND parameter allows you to specify the condition for executing a job step.

The compiler, assembler, and linkage editor programs issue return codes. You may want to use the COND parameter to test these return codes. If you write your processing programs in assembler language, ANS COBOL, FORTRAN, or PL/I, you can use the COND parameter to test return codes issued by your programs.

BYPASSING A JOB STEP

The return code tests specified in the COND parameter determine whether a job step is to be bypassed. Each return code test consists of a code, an operator, and, optionally, a stepname. The operator indicates the mathematical relationship between the code specified on the EXEC statement and the code returned by a completed job step. The operator or operators are compared with the return code or codes and if any of the relationships are true, the job step is bypassed.

If the return code test includes a stepname, the test is made using the return code issued by the named step. If the named step was not executed, the request for a test is ignored. If the return code test does not include a stepname, the test is made using the return code issued by every preceding job step that completed normally. To test in a later job step the return code issued by a cataloged procedure step, specify both the name of the job step that called the procedure and the procedure stepname, i.e., stepname.procstepname.

EXECUTING A JOB STEP

Abnormal termination of a job step normally causes subsequent steps to be bypassed and the job to be terminated. By means of the COND parameter, you can specify the condition for executing a job step after one or more of the preceding job steps have abnormally terminated. For the COND parameter, a job step is considered to abnormally terminate if a failure occurs within the user's program once it has received control. (If, during scheduling, a job step is not scheduled for execution because of failures such as job control language errors or inability to allocate space, the remainder of the job steps are bypassed, whether or not a condition for executing a later job step was specified.)

The condition for executing a job step after one or more of the preceding job steps have abnormally terminated is either EVEN or ONLY. EVEN causes the step to be executed even if one or more of the preceding job steps have abnormally terminated; ONLY causes the step to be executed only if one or more of the preceding job steps have abnormally terminated. When a job step abnormally terminates, the COND parameter on the EXEC statement of the next step is scanned for the EVEN or ONLY subparameter. If neither is specified, the next job step is bypassed and the EXEC statement of the following step is scanned for EVEN or ONLY. If

EVEN or ONLY is specified, return code tests, if any, are made on all previous steps specified that did not abnormally terminate. The step is bypassed if any one of these tests is satisfied, or if one of the previous job steps abended because it exceeded the time limit for the job. Otherwise, the job step is executed.

Caution: When a job step that contains the EVEN or ONLY subparameter refers to a data set that was to be created or cataloged in a preceding step, the data set (1) will not exist if the step creating it was bypassed, or (2) may be incomplete if the step creating it abnormally terminated. Also, if the job step refers the system to an earlier job step for volume and unit information, this information is not available if the earlier job step was bypassed. If ONLY is specified on the first job step and a JOBLIB is being used, the unit and volume information are not passed to the succeeding step and the catalog will be searched for the JOBLIB data set.

WHEN YOU CALL A CATALOGED PROCEDURE

The COND parameter may be coded on the EXEC statement of a cataloged procedure step. If the job step calls a cataloged procedure, you may want to override all COND parameters in the procedure or only certain COND parameters. To override all COND parameters, code the COND parameter on the EXEC statement that calls the procedure. This establishes one set of return code tests and the EVEN or ONLY subparameter for all steps in the procedure. To override only certain COND parameters, code, on the EXEC statement that calls the procedure, COND.procstepname for each procedure step that you want to override. Return code tests and the EVEN or ONLY subparameter will then pertain only to the named procedure step. When the condition parameter appears on both JOB and EXEC statements, the conditions on the JOB card override those on the EXEC.

Examples of the COND Parameter

1. `//STEP6  EXEC PGM=BAB,COND=(4,GT,STEP3)`

   If 4 is greater than the return code issued by STEP3, this step is bypassed. (A return code of 4 or greater from STEP3 allows this step (STEP6) to be executed.) If STEP3 was not executed, however the request for a test is ignored. Since neither EVEN nor ONLY is specified, this job step is automatically bypassed if a preceding step abnormally terminates.

2. `//TEST2  EXEC PGM=BACK,COND=((16,GE),(90,LE,STEP1),ONLY)`

   If 16 is greater than or equal to the return code issued by any of the preceding job steps or if 90 is less than or equal to the return code issued by STEP1, this step is bypassed. If none of the tests are satisfied and a preceding job step has abnormally terminated, this step is executed because ONLY is coded.

3. `//PRCH  EXEC PGM=SPE,COND=(12,EQ,STEP4.LOOKUP)`

   If 12 is equal to the return code issued by the procedure step named LOOKUP, the job step is bypassed. Since neither EVEN nor ONLY is specified, this job step would be automatically bypassed if a preceding step abnormally terminated.

4.
```
//STP4 EXEC BILLING,COND.PAID=(EVEN,(20,LT)),                    X
//          COND.LATE=(60,GT,FIND),COND.BILL=((20,GE),(30,LT,CHGE))
```

Specifies that different return code tests pertain to each of the named cataloged or in-stream procedure steps (PAID, LATE, and BILL). If the return code test specified for the procedure step named PAID is not satisfied, the step is executed even if a preceding step abnormally terminated.

The DPRTY Parameter (For MVT)

```
DPRTY=(value1,value2)
```

value1
a number from 0 through 15. If you do not assign a number, a value of 0 is assumed.

value2
a number from 0 through 15. If you do not assign a number, a value of 11 is assumed.

Rules for Coding

1. Avoid assigning a number of 15 to value1. This number is used for certain system tasks.

2. If you omit value2, you need not code the parentheses.

3. If you omit value1, you must code a comma preceding value2 to indicate the absence of value1.

4. If the DPRTY parameter is coded for MFT, the parameter is not used, but is checked for syntax.

Assigning a Dispatching Priority

The DPRTY parameter is used to assign a dispatching priority to a job step. Dispatching priority determines in what order tasks will use main storage and CPU resources. If you do not code the DPRTY parameter, the job step is assigned the priority assigned to the job either on the JOB statement (the PRTY parameter) or by default.

Value1 of the DPRTY parameter has the same meaning as the value you assign in the PRTY parameter on the JOB statement. That is, if you code PRTY=10 on the JOB statement and DPRTY=10 on the EXEC statement, the job and step priority are the same. Also, in this case the job and step have the same dispatching priority. This is because the system converts the number 10 to an internal priority and then adds 11 to the internal priority to form the dispatching priority (11 is always the number added to the job's internal priority; 11 is the number added to the job step's internal priority when value2 of the DPRTY parameter is omitted).

If you code value2 of the DPRTY parameter, the system adds that value to the internal priority to form the dispatching priority. (The internal priority is formed by the system by converting the value assigned to value1 in the DPRTY parameter.)

When you want the job step to have a different dispatching priority than the job, you code the DPRTY parameter and either raise or lower the values, depending on whether the step is to have a higher or lower priority than the job.

If the DPRTY parameter specifies a dispatching priority greater than the initiator's, the initiator's priority will be used.

THE DPRTY PARAMETER AND TIME-SLICING

If your installation provides time-slicing facilities in a system with MVT, the DPRTY parameter can be used to make a job step part of a group of jobs and job steps to be time-sliced. (To make an entire job part of a group of jobs and job steps to be time-sliced, code the PRTY parameter on the JOB statement.) At system generation, the priorities of the time-sliced groups are selected. If the number assigned to "value1" corresponds to a priority number selected for time-slicing and "value2" is either omitted or assigned a value of 11, then the job step's tasks will be time-sliced.

WHEN YOU CALL A CATALOGED PROCEDURE

The DPRTY parameter may be coded on the EXEC statement of a cataloged procedure step. If the job step calls a cataloged procedure, you may want to override all DPRTY parameters in the procedure or only certain DPRTY parameters. To override all DPRTY parameters, code the DPRTY parameter on the EXEC statement that calls the procedure. This establishes one dispatching priority for all the steps in the procedure. To override only certain DPRTY parameters, code, on the EXEC statement that calls the procedure, DPRTY.procstepname for each procedure step that you want to override. The dispatching priority will then pertain only to the named procedure step.

Examples of the DPRTY Parameter

1. `//BP2 EXEC PGM=FOUR,DPRTY=(13,9)`

   The system uses these numbers to form a dispatching priority for this step. Since the numbers are high, the dispatching priority will be high.

2. `//STEP3 EXEC PGM=BROWN31,DPRTY=(,12)`

   The system first assigns a value of 0 to the absent subparameter and then forms a dispatching priority. In this case, the dispatching priority will be very low.

3. `//ST2 EXEC COMP,DPRTY=4`

   The system assigns a dispatching priority of 4 to all steps in the procedure named COMP.

The PARM Parameter

PARM=value

value
consists of up to 100 characters of information or options that the system is to pass to the processing program.

Rules for Coding

1. If the value contains more than one expression separated by commas, the value must be enclosed in apostrophes or parentheses, e.g., PARM='P1,123,MT5' or PARM=(P1,123,MT5). (Enclosing apostrophes and parentheses are not passed to the processing program; commas within apostrophes and parentheses are passed as part of the value.)

2. If any expression contains special characters, either (1) enclose the value in apostrophes, or (2) enclose the expression in apostrophes and the value in parentheses, e.g., PARM='P50,12+80' or PARM=(P50,'12+80'). (The enclosing apostrophes and parentheses are not considered part of the value.) If one of the special characters is an apostrophe, code two consecutive apostrophes in its place, e.g., PARM='CONTROL INFORM''N'. If one of the special characters is an ampersand and you are not defining a symbolic parameter, code two consecutive ampersands in its place, e.g., PARM='3462&&5'. (When two apostrophes or two ampersands are coded, only one is passed to the processing program.)

3. If the value must be continued on another statement, enclose the value in parentheses. The continuation comma is considered part of the value field and counts towards the maximum of 100 characters of data. You may not continue on another statement any value enclosed in apostrophes.

Providing a Processing Program With Information at Execution Time

Some information required by a program may vary from application to application, such as module attributes and options required by compiler, assembler, and linkage editor programs. In order to provide this information to the program at the time it is executed, you can code the PARM keyword parameter. The program must include instructions that can retrieve this information. (The exact location and format of the information passed to a processing program are described in Supervisor Services and Macro Instructions.

WHEN YOU CALL A CATALOGED OR IN-STREAM PROCEDURE

The PARM parameter may be coded on the EXEC statement of a cataloged or in-stream procedure step. If the job step calls a cataloged or in-stream procedure, you can pass information to the first procedure step and nullify all other PARM parameters in the procedure or override some of the PARM parameters contained in the procedure. To accomplish the first, code the PARM parameter on the EXEC statement that calls the procedure. The information contained in the PARM parameter is passed to the first procedure step and PARM parameters in all other procedure steps are nullified. To override some of the PARM parameters contained in the procedure, code, on the EXEC statement that calls the procedure, PARM.procstepname for each procedure step that you want to override. Information provided is passed only to the named procedure step.

Examples of the PARM Parameter

1. `//RUN3  EXEC PGM=APG22,PARM=(P1,123,'P2=5')`

   The system passes the information in the PARM parameter, except the apostrophes, to the processing program named APG22.

2. `//  EXEC PROC81,PARM=MT5`

   The system passes this information to the first step of the procedure named PROC81. If any of the other procedure steps contain the PARM parameter, these parameters are nullified.

3. `//STP6  EXEC ASMFCLG,PARM.LKED=(MAP,LET)`

   The system passes this information to the procedure step named LKED. If any of the other procedure steps contain the PARM parameter, these parameters are still in effect.

The RD Parameter

```
RD=(R   )
   (RNC )
   (NC  )
   (NR  )
```

R
specifies that automatic step restart is permitted.

RNC
specifies that automatic step restart is permitted and automatic checkpoint restart is not permitted and no checkpoints can be established.

NC
specifies that neither automatic step restart nor automatic checkpoint restart is permitted and no checkpoints can be established.

NR
specifies that neither automatic step restart nor automatic checkpoint restart is permitted, but the CHKPT macro instruction can establish a checkpoint.

Rules for Coding

1. Be sure to code MSGLEVEL=(1,1), MSGLEVEL=(1,0), or MSGLEVEL=1 when RD=R or RD=RNC is specified.

2. If you are permitting automatic step restart, assign the step a unique step name.

3. If you have coded the RD parameter on the JOB statement, RD parameters on the job's EXEC statements are ignored.

Using the Restart Facilities

The RD (restart definition) keyword parameter is coded when you want to make use of the step restart facilities, to suppress the action of the CHKPT macro instruction, or to suppress automatic restarts. The step restart facilities permit execution of a job to be automatically restarted at a job step after the job abnormally terminates or after a system failure occurs. Through the RD parameter, you can specify that execution of a job step is to be automatically restarted at the beginning of the step if it abnormally terminates (step restart).

Execution of a job step can also be automatically restarted within the step if it abnormally terminates (checkpoint restart). In order for checkpoint restart to occur, the CHKPT macro instruction must have been executed in the processing program before abnormal termination. When you use the RD parameter to request suppression of the CHKPT macro instruction action, automatic checkpoint restart cannot occur.

If the RD parameter is not coded, step restart cannot occur. If the RD parameter is not coded and the processing program contains CHKPT macro instructions, checkpoint restart can occur.

The following three conditions must be met before automatic step or checkpoint restart can occur: (1) the completion code returned during abnormal termination indicates that the step is eligible for restart, (2) the operator authorizes restart, and (3) MSGLEVEL=(1,0),

MSGLEVEL=(1,1), or MSGLEVEL=1 must be coded on the JOB statement. If these conditions are satisfied, special disposition processing is performed before restart. If automatic step restart is to occur, all data sets in the restart step with a status of OLD or MOD, and all data sets being passed to steps following the restart step, are kept. All data sets in the restart step with a status of NEW are deleted. If automatic checkpoint restart is to occur, all data sets currently in use by the job are kept.

DEFINING RESTART

You define the type of restart that can occur by coding one of the subparameters of the RD parameter: R, RNC, NC, or NR. Each of these subparameters is described in detail in the following paragraphs.

RD=R: R indicates that automatic step restart is permitted. If the processing program used by the job step does not include any CHKPT macro instructions, coding RD=R allows execution to be resumed at the beginning of this step if it abnormally terminates. If the program does include a CHKPT macro instruction, coding RD=R permits automatic step restart to occur only if the step abnormally terminates before execution of the CHKPT macro instruction; thereafter, only checkpoint restart can occur. If you cancel the effects of the CHKPT macro instruction before a checkpoint restart is performed, the request for automatic step restart is again in effect.

RD=RNC: RNC indicates that automatic step restart is permitted and automatic checkpoint restart is not permitted. RD=RNC should be specified when you want to suppress the action of all CHKPT macro instructions included in the processing program and to permit automatic step restart.

RD=NC: NC indicates that neither automatic step restart nor automatic checkpoint restart is permitted. RD=NC should be specified when you want to suppress the action of all CHKPT macro instructions included in the processing program and not to permit automatic step restart. RD=NC has no effect on processing if CHKPT macro instructions are not included in the program.

RD=NR: NR indicates that a CHKPT macro instruction can establish a checkpoint, but neither automatic step restart nor automatic checkpoint restart is permitted. Coding RD=NR allows you to resubmit the job at a later time and specify in the RESTART parameter the checkpoint at which execution is to be resumed. (The RESTART parameter is coded on the JOB statement of the resubmitted job.) RD=NR has no effect on processing if CHKPT macro instructions are not included in the program.

WHEN YOU CALL A CATALOGED PROCEDURE

The RD parameter may be coded on the EXEC statement of a cataloged procedure step. If the job step calls a cataloged procedure, you may want to override all RD parameters in the procedure or only certain RD parameters. To override all RD parameters, code the RD parameter on the EXEC statement that calls the procedure. This establishes one restart request for all the steps in the procedure. To override only certain RD parameters, code, on the EXEC statement that calls the procedure, RD.procstepname for each procedure step that you want to override. The restart request will then pertain only to the named procedure step.

References

1. For detailed information on the checkpoint/restart facilities, refer to the publication Advanced Checkpoint/Restart Planning Guide, the topic "Checkpoint and Restart" in the publication Supervisor Services and Macro Instructions, and "Using the Restart Facilities" in Appendix B of this publication.

2. For information on how to code the CHKPT macro instruction, refer to the publication Supervisor Services and Macro Instructions.

Examples of the RD Parameter

1. `//STEP1  EXEC PGM=GIIM,RD=R`

   Permits execution to be automatically restarted with this step if it abnormally terminates.

2. `//NEST  EXEC PGM=T18,RD=RNC`

   Permits execution to be automatically restarted with this step if it abnormally terminates; suppresses the action of CHKPT macro instructions issued in the program this job step uses.

3. `//CARD  EXEC PGM=WTR,RD=NR`

   Neither automatic step restart nor automatic checkpoint restart can occur, but CHKPT macro instructions issued in the program that this job step executes can establish checkpoints.

4. `//STP4  EXEC BILLING,RD.PAID=NO,RD.BILL=NR`

   Specifies that different restart requests pertain to each of the named procedure steps (PAID and BILL).

The REGION Parameter - Without Main Storage Hierarchy Support (For MVT)

```
REGION=valueK
```

valueK
specifies the number of contiguous 1024-byte areas of main storage to be allocated to the job step. The number can range from one to five digits but may not exceed 16383.

Rules for Coding

1. Code an even number. (If you code an odd number, the system treats it as the next highest even number. When the value 16383K is coded, the system treats it as 16384K. However, the value 16384K must not be coded on the EXEC statement.)

2. If you have coded the REGION parameter on the JOB statement, REGION parameters on the job's EXEC statements are ignored.

3. If the REGION parameter is coded for MFT, the parameter is not used, but is checked for syntax.

Requesting Main Storage

The REGION keyword parameter is used to specify how much main storage, in contiguous bytes, is to be allocated to the job step. Code the REGION parameter when you want more storage or less storage than would be allocated if the default region size was used. The default region size is established as a PARM parameter field in the cataloged procedure for the input reader. You can consult the Storage Estimates publication to help you determine how much main storage is required to process your job.

ACQUIRING ADDITIONAL MAIN STORAGE

If the step may require use of more main storage than has been allocated, you can code the ROLL parameter on either the JOB statement or EXEC statement and request that the system try to provide you with additional main storage. The ROLL parameter is described in the chapters "The ROLL Parameter" later in this section and in Section II.

WHEN YOU CALL A CATALOGED PROCEDURE

The REGION parameter may be coded on the EXEC statement of a cataloged procedure step. If the job step calls a cataloged procedure, you may want to override all REGION parameters in the procedure or only certain REGION parameters. To override all REGION parameters, code the REGION parameter on the EXEC statement that calls the procedure. Each procedure step will be allocated the same amount of storage. To override only certain REGION parameters, code, on the EXEC statement that calls the procedure, REGION.procstepname for each procedure step that you want to override. The requested region size will then be allocated only to the named procedure step.

Examples of the REGION Parameter

1. `//JUNE  EXEC PGM=A1403,REGION=112K`

   Specifies that 112 contiguous 1024-byte areas of main storage are to be allocated to the job step.

2. `//STP2  EXEC PGM=RATL,REGION=70K,ROLL=(YES,YES)`

   The REGION parameter specifies that 70 contiguous 1024-byte areas of main storage are to be allocated to the job step. In the ROLL parameter, the first subparameter tells the system that this step may be rolled out if additional storage is required by another job; the second subparameter tells the system that it should try to provide this step with additional main storage if it is required.

3. `//STP4  EXEC BILLING,REGION.LATE=80K,REGION.BILL=108K`

   Specifies that different region sizes are to be allocated to the named procedure steps (LATE and BILL).

The REGION Parameter - With Main Storage Hierarchy Support (For MVT, Excluding M65MP)

REGION=(valueK,value$_1$K)

valueK
specifies the number of contiguous 1024-byte areas in hierarchy 0 to be allocated to the job step. If IBM 2361 Core Storage is present, the number cannot exceed 16383.

value$_1$K
specifies the number of contiguous 1024-byte areas in hierarchy 1 to be allocated to the job step. If IBM 2361 Core Storage is present, the number cannot exceed 1024 (for each Model 1) or 2048 (for each Model 2).

Rules for Coding

1. When processor storage includes hierarchies 0 and 1, the sum of value and value$_1$ cannot exceed 16383.

2. Code even numbers. (If you code an odd number, the system treats it as the next highest even number. When 16383K is coded the system treats it as 16384K. However, a sum of 16384K must not be coded on the EXEC statement.)

3. When you are requesting storage only in hierarchy 1, precede value$_1$ with a comma, to indicate the absence of value.

4. When you are requesting storage only in hierarchy 0, you need not code the parentheses.

5. If you have coded the REGION parameter on the JOB statement, REGION parameters on the job's EXEC statements are ignored.

6. If the REGION parameter is coded for MFT, the parameter is not used, but is checked for syntax.

Requesting Main Storage in One or Two Hierarchies

The REGION keyword parameter is used to specify how much main storage is to be allocated to each job step, and, when main storage hierarchy support has been specified at system generation, in which hierarchy or hierarchies to allocate main storage. With main storage hierarchy support, storage hierarchies 0 and 1 are provided. If IBM 2361 Core Storage, Model 1 or 2, is present in the system, processor storage is referred to as hierarchy 0 and 2361 Core Storage is referred to as hierarchy 1. If 2361 Core Storage is not present, a two-part region is established in processor storage when regions are requested in two hierarchies. The two parts are not necessarily contiguous in processor storage.

Code the REGION parameter to specify how much storage is to be allocated in each hierarchy, or that all storage for the job step is to be allocated in a particular hierarchy. (If you do not code the REGION parameter on either the JOB or EXEC statement, the default region size, which is a PARM parameter field in the cataloged procedure for the input reader, is used and is always allocated in hierarchy 0. If you code the REGION parameter and request storage only from hierarchy 1, no hierarchy 0 segment will be allocated.) You can consult the Storage Estimates publication to help you determine how much main storage is required to process the job step. Then, depending on your reasons for using hierarchies, determine how much storage is required in each.

If main storage hierarchy support was not specified at system generation and regions are requested in both hierarchies, the region sizes are combined and an attempt is made to allocate a single region from processor storage. If a region is requested entirely from hierarchy 1, an attempt is made to allocate the region from processor storage.

ACQUIRING ADDITIONIAL MAIN STORAGE

If the job step may require more main storage than has been allocated, you can code the ROLL parameter and request that the system try to provide you with additional main storage in that hierarchy. The ROLL parameter is described in the chapters "The ROLL Parameter" later in this section and in Section II.

WHEN YOU CALL A CATALOGED PROCEDURE

REGION parameter may be coded on the EXEC statement of a cataloged or in-stream procedure step. If the job step calls a cataloged procedure, you may want to override all REGION parameters in the procedure or only certain REGION parameters. To override all REGION parameters, code the REGION parameter on the EXEC statement that calls the procedure. Each procedure step will be allocated the same amount of storage in the specified hierarchies. To override only certain REGION parameters, code, on the EXEC statement that calls the procedure, REGION.procstepname for each procedure step you want to override. The requested region size will then be allocated in the specified hierarchies only to the named procedure step.

Examples of the REGION Parameter

1. `//MART  EXEC PGM=TYP,REGION=(80K,30K)`

   Specifies that the system is to allocate 80 contiguous 1024-byte areas of storage in hierarchy 0 and 30 contiguous 1024-byte areas of storage in hierarchy 1. If main storage hierarchy support is not included in the system, the system will try to obtain 110 contiguous 1024-byte areas in processor storage.

2. `//  EXEC PGM=U1489,REGION=(98K)`

   Specifies that the system is to allocate 98 contiguous 1024-byte areas of storage in hierarchy 0.

3. `//RAND  EXEC PGM=SSYS,REGION=(100K,50K),ROLL=(YES,YES)`

   The REGION parameter specifies that the system is to allocate 100 contiguous 1024-byte areas of storage in hierarchy 0 and 50 contiguous 1024-byte areas of storage in hierarchy 1. In the ROLL parameter, the first subparameter tells the system that this step may be rolled out if additional storage is required by another job; the second subparameter tells the system that it should try to provide this step with additional main storage if it is required.

4. `//STP4  EXEC BILLING,REGION.PAID=(28K,10K),REGION.LATE=(44K,8K)`

named procedure steps (PAID and LATE).

   Specifies that different region sizes are to be allocated to the named proceudre steps (PAID and LATE).

The ROLL Parameter (For MVT)

ROLL=(x,y)

x
declares whether the job step may be rolled out. Code YES if the step may be rolled out; code NO if the step may not be rolled out.

y
declares whether the job step may cause rollout of another job step. Code YES if the step may cause rollout of another job step; code NO if the step may not cause rollout of another job step. YES must be coded if you want additional main storage allocated to the step when additional main storage is required.

Rules for Coding

1. If you code the ROLL parameter, both subparameters must be specified.
2. If you have coded the ROLL parameter on the JOB statement, ROLL parameters coded on the job's EXEC statements are ignored.
3. Code ROLL=(NO,YES) or ROLL=(NO,NO) if this step is part of a teleprocessing job that uses the Auto Poll option. If you allow the step to be rolled out, the step cannot be restarted properly.
4. If the ROLL parameter is coded for MFT, the parameter is not used, but is checked for syntax.

When to Code the ROLL Parameter

The ROLL keyword parameter should be coded if the job step may require more main storage than was requested in the REGION parameter. When you specify in the ROLL parameter that this job step may cause rollout of another job step, an attempt is made to allocate additional storage if the step requires it. In order to allocate this additional space to a job step, another job step with a lower priority may have to be rolled out, i.e., temporarily transferred to secondary storage.

The ROLL parameter should also be coded when you want control over whether the job step can be rolled out because of another step's need for additional main storage. If the ROLL parameter is not coded, the specification made in the PARM parameter field in the cataloged procedure for the input reader is used.

WHEN YOU CALL A CATALOGED PROCEDURE

The ROLL parameter may be coded on the EXEC statement of a cataloged procedure step. If the job step calls a cataloged procedure, you may want to override all ROLL parameters in the procedure or only certain ROLL parameters. To override all ROLL parameters, code the ROLL parameter on the EXEC statement that calls the procedure. This establishes one rollout/rollin request for all the steps in the procedure. To override only certain ROLL parameters, code, on the EXEC statement that calls the procedure, ROLL.procstepname for each procedure step that you want to override. The rollout/rollin request will then pertain only to the named procedure step.

Examples of the ROLL Parameter

1. `//FILL EXEC PGM=PLUS,ROLL=(YES,YES),REGION=100K`

   Specifies that this step may be rolled out and may cause rollout of another job step if this step requires more than 100K of main storage.

2. `//UP EXEC PGM=2165,ROLL=(NO,YES)`

   Specifies that this step may not be rolled out but may cause rollout of another job step.

3. `//STP4 EXEC BILLING,ROLL.LATE=(YES,NO),ROLL.BILL=(NO,NO)`

   Specifies that different rollout/rollin requests pertain to each of the named procedure steps (LATE and BILL).

The TIME Parameter

```
TIME=    {(minutes,seconds)}
         {1440             }
```

minutes
: specifies the maximum number of minutes the job step can use the CPU. The number of minutes must be less than 1440 (24 hours).

seconds
: specifies the maximum number of seconds beyond the specified number of minutes the job step can use the CPU, or, if no minutes are specified, the maximum number of seconds the job step can use the CPU. The number of seconds must be less than 60.

1440
: specifies that the job step is not to be timed. Code 1440 if the step may require use of the CPU for 24 hours or more or if the step should be allowed to remain in a wait state for more than the established time limit.

Rules for Coding

1. If the CPU time limit is given in minutes only, you need not code the parentheses.

2. If the CPU time limit is given in seconds only, you must code a comma preceding the seconds to indicate the absence of minutes.

3. You must not code TIME=0 on an EXEC statement.

Specifying a Time Limit for a Job Step

The TIME keyword parameter can be used to specify the maximum amount of time the job step may use the CPU. (CPU time will appear on the output listing if system management facilities (SMF) or users accounting routines supply this information to the output data set.) By coding the TIME parameter, you can limit the CPU time wasted by a step that goes into a loop. Normally, a step that exceeds the specified time limit causes termination of the job. However, if the System Management Facilities option is included in the system and a user exit routine is provided, this routine can extend the time limit so that processing can continue. When the TIME parameter is not coded, a default time limit is assumed. The default is specified as a PARM parameter field in the cataloged procedure for the input reader.

TIME LIMIT FOR WAIT STATES

Since the job step can go into an extremely long wait state, the time a job step may remain in a wait state is limited. If the System Management Facilities option is included in the system, the installation determines this time limit. In this case, if the job step remains in a wait state for more than the established time limit, the job is terminated unless a user-provided exit routine extends the wait-state time limit for the step. If the System Management Facilities option is not included, the system automatically provides a 30-minute time limit for wait states; if the job step remains in a wait state for more than 30 consecutive minutes, the job is terminated.

How to Eliminate Timing

Certain applications require that a job use the CPU for 24 hours or more. In these cases you must eliminate job and step timing by coding TIME=1440. You should also code TIME=1440 if any of the job's steps should be allowed to remain in a wait state for more than the established time limit.

If your system includes the System Management Facilities (SMF) feature and you code TIME=1440, SMF termination messages will indicate that no CPU time was used; messages indicating the time the job step started and stopped will not, however, be affected.

HOW THE JOB TIME LIMIT AFFECTS THE STEP TIME LIMIT

The remaining job time may affect the amount of time the step can use the CPU. If the remaining CPU time for the job is less than the CPU time limit specified on the EXEC statement, the step can use the CPU only for the job's remaining CPU time. For example, if the job's remaining CPU time is 5 minutes and the step specifies a CPU time limit of 10 minutes, the step can only use the CPU for 5 minutes.

WHEN YOU CALL A CATALOGED PROCEDURE

The TIME parameter may be coded on the EXEC statement of a cataloged (or an in-stream) procedure step. If the job step calls a cataloged procedure, you may want to override all TIME parameters in the procedure or only certain TIME parameters. To override all TIME parameters, code the TIME parameter on the EXEC statement that calls the procedure. This applies a CPU time limit for the entire procedure, and nullifies any TIME parameters that appear on EXEC statements in the procedure. To override only certain TIME parameters, code, on the EXEC statement that calls the procedure, TIME.procstepname for each procedure step that you want to override. The CPU time limit will then pertain only to the named procedure step.

Reference

1. A discussion of the System Management Facilities option is contained in the Introduction publication. Information on user exit routines to be used with the System Management Facilities option is contained in the System Management Facilities Guide.

Examples of the TIME Parameter

1. `//STEP1  EXEC PGM=GRYS,TIME=(12,10)`

   Specifies that the maximum amount of time the step can use the CPU is 12 minutes 10 seconds.

2. `//FOUR  EXEC PGM=JPLUS,TIME=(,30)`

   Specifies that the maximum amount of time the step can use the CPU is 30 seconds.

3. `//INT  EXEC PGM=CALC,TIME=5`

   Specifies that the maximum amount of time the step can use the CPU is 5 minutes.

4. `//LONG  EXEC PGM=INVANL,TIME=1440`

   Specifies that the job step is not to be timed. Therefore, the step may use the CPU and may remain in a wait state for an unspecified period of time.

5. `//STP4  EXEC BILLING,TIME.PAID=(45,30),TIME.BILL=(112,59)`

   Specifies that different time limits pertain to each of the named procedure steps.

The DD (data definition) statement describes a data set that is to be used in a job step and specifies the input and output facilities required for use of the data set. Each data set to be used in a step requires a DD statement; all DD statements for a step follow that step's EXEC statement. Although all DD statement parameters are optional, a blank operand field is invalid, except when you are overriding DD statements that define concatenated data sets. (See "Overriding DD Statements that Define Concatenated Data Sets" in Appendix A of this publication.) You can include a maximum of 255 DD statements per job step.

DD Statement Format

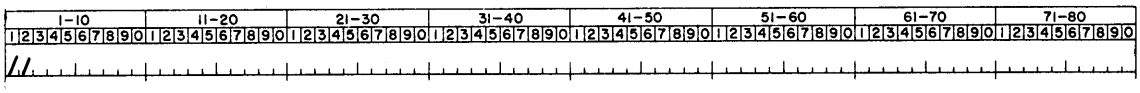
```
//ddname  DD  operands  comments
```

The DD statement consists of the characters //, in columns 1 and 2, and four fields - the name, operation (DD), operand, and comments field.

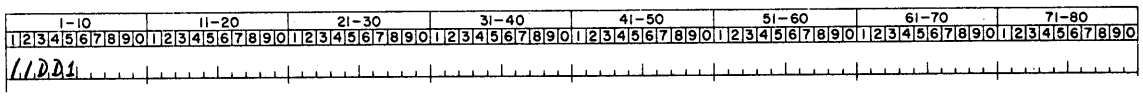
Rules for Coding

Follow the order listed below when coding the DD statement:

1. Code the characters // in columns 1 and 2.

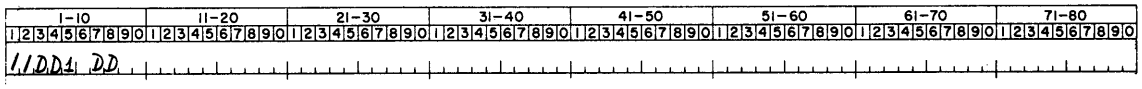


2. Code a ddname, starting in column 3. (A ddname is not coded in two cases. These cases are described in the chapter "Assigning a Ddname.")



3. Follow the ddname, or // if a ddname is not coded, with at least one blank.

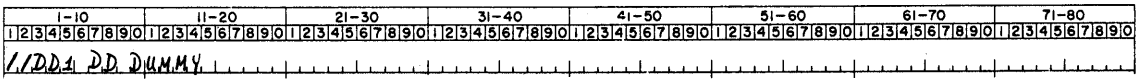
4. Code DD.



5. Follow DD with at least one blank.

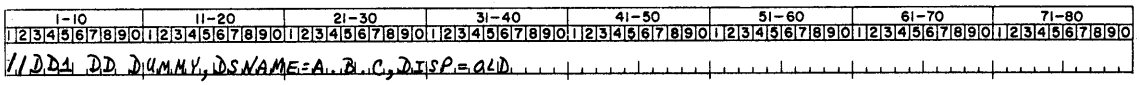
6. Code any desired positional parameter.

7. Code any desired keyword parameters. Separate each parameter with a comma.



8. Code at least one blank.

9. Code any desired comments.



Positional and Keyword Parameters

There are two types of parameters that can be coded on the DD statement:

Positional parameters, which must precede any keyword parameters. One of the following positional parameters may be coded on a DD statement:

*

DATA

DUMMY

DYNAM

These positional parameters are described in the following pages in the order listed above.

Keyword parameters, which may be coded in any order. The following keyword parameters can be coded on a DD statement:

AFF

DCB

DDNAME

DISP

DLM

DSN (see DSNAME)

DSNAME

FCB

LABEL

OUTLIM

QNAME - MFT and MVT with TCAM

SEP

SPACE

SPLIT

SUBALLOC

SYSOUT

TERM - MVT with TSO

UCS

UNIT

VOL (see VOLUME)

VOLUME

These keyword parameters are described, after the positional parameters, in the order listed above.

Sample DD Statements

```
1.  //DDA      DD   DSNAME=&&TEMP,UNIT=2400,DISP=(NEW,PASS)

2.  //PRINT    DD   SYSOUT=F

3.  //IN       DD   DSNAME=ALLOC,DISP=(,KEEP,DELETE),UNIT=2311,          X
    //                VOLUME=SER=541382,SPACE=(CYL,(12,1))

4.  //DWN      DD   *
```

Assigning a Ddname

```
//ddname DD
```

The ddname identifies a DD statement so that subsequent control statements and the data control block in the processing program can refer to it. The ddname must begin in column 3 and consist of 1 through 8 alphameric or national (@, #, $) characters. The first character must be an alphabetic or national character.

Each ddname within a job step should be unique. If duplicate ddnames exist between successive EXEC statements within one job, allocation of devices and space and disposition processing are done for both DD statements; however, all references are directed to the first such DD statement in the step.

There are several special ddnames that tell the system that you want to make use of particular facilities. Except for the ddname SYSCHK, do not use the special ddnames unless you want these facilities. These special ddnames are individually discussed following "Examples of Valid Ddnames" in the section titled "Special Ddnames".

Apart from the restricted use of certain special ddnames, there are two instances when you should not code a ddname at all:

1. If a DD statement is to define a data set that is concatenated with a data set defined by a preceding DD statement.

2. If the DD statement is the second or third consecutive DD statement that defines an indexed sequential data set. (Defining an indexed sequential data set on more than one DD statement is discussed in "Appendix C: Creating and Retrieving Indexed Sequential Data Sets.")

WHEN ADDING OR OVERRIDING INFORMATION IN A CATALOGED PROCEDURE STEP

If the job step uses a cataloged or a in-stream procedure, DD statements that follow the EXEC statement are used (1) to override parameters on the various DD statements in the procedure, and (2) to add new DD statements to the procedure. These modifications exist only for the duration of the job step; they do not change the procedure permanently.

To make one of these modifications, each ddname must be qualified by a procedure step name, i.e., procstepname.ddname,as follows:

1. To override parameters on a DD statement, code the name of the procedure step in which the DD statement appears, followed by a period, followed by the name of the DD statement that you want to override.

2. To add DD statements to a procedure step, code the name of the procedure step in which you want to add the statement, followed by a period, followed by a ddname of your choosing.

3. To supply a procedure step with data in the input stream, code the name of the procedure step that is to use the data, followed by a ddname. This ddname may be predefined in the procedure step by means of the DDNAME parameter. In this case, the ddname that follows the procedure step name is the name coded in the DDNAME parameter. Otherwise, you code a ddname of your choosing.

4. To define data sets that are to be concatenated and added to the procedure step, identify the procedure step in which you want to add the statements, follow with a ddname of your choosing on the first DD statement, and omit the ddname on the second DD statement.

Examples of Valid Ddnames

1. `//DD1 DD`

2. `//#5863 DD`

3. `//INPUT DD`
   `// DD`

   Because the ddname is missing from the second DD statement, the data sets defined in these statements are concatenated.

4. `//PAYROLL.DAY DD`

   If the procedure step named PAYROLL includes a DD statement named DAY, this statement overrides parameters on the statement named DAY. If the step does not include a DD statement named DAY, this statement is added to the procedure step for the duration of the job step.

5. `//STEPSIX.DD4 DD`
   `// DD`

   By identifying the procedure step in which you want to add statements (STEPSIX), followed by a ddname of your choosing (DD4), you can define data sets that are to be concatentated and added to the procedure step.

Special Ddnames

There are five special ddnames that tell the system you want to make use of a particular facility. The five ddnames and their functions are:

- JOBLIB - this DD statement defines a private library that the system makes available for use by the job.
- STEPLIB - this DD statement defines a private library that the system makes available for use by a job step.
- SYSABEND - this DD statement defines a data set on which a dump can be written if the step abnormally terminates. The dump provided would include the system nucleus, the processing program storage area, and, possibly, a trace table.
- SYSUDUMP - this DD statement defines a data set on which a dump can be written if the step abnormally terminates. The dump provided would include only the processing program storage area.
- SYSCHK - this DD statement defines the checkpoint data set and is included when a deferred checkpoint restart is to occur.

JOBLIB

Unless the system is told that the program you request on the EXEC statement resides in a private or temporary library, the system expects to find it in the system library (SYS1.LINKLIB). One way to tell the system that a program resides in a private library is to follow the JOB statement with a DD statement named JOBLIB. (The other way to tell the system that a program resides in a private library is to include, as one of the DD statements for a job step, a DD statement named STEPLIB. The STEPLIB DD statement is described under the next topic, "STEPLIB.") If you include a JOBLIB DD statement, each time you request a program the system first looks in the private library; if the system does not find the program there, the system looks for it in the system library.

The parameters you code on the JOBLIB DD statement are determined by whether the library is cataloged. The parameters that must be coded when the library is cataloged and when the library is not cataloged are described under "When the Library Is Cataloged" and "When the Library Is Not Cataloged," respectively. In either case, how you code the DISP parameter is the same and is described in the topic "The DISP Parameter."

RULES FOR CODING THE JOBLIB DD STATEMENT

1. The ddname must be JOBLIB. Never use the ddname JOBLIB except when you are defining a private library.
2. The JOBLIB DD statement must appear immediately after the JOB statement to which it pertains.
3. A JOBLIB DD statement cannot appear in a cataloged procedure.

The DISP Parameter

To make the private library available throughout the job, you must code the DISP parameter to specify the library's status and disposition. One of the following may be coded:

1. DISP=(OLD,PASS)

   The library already exists and is kept at the end of the job. If you code DISP=OLD, the system assumes DISP=(OLD,PASS).

2. DISP=(SHR,PASS)

   The library already exists and is kept at the end of the job. The library may be used by other jobs that are currently being executed, as long as all references to the library within the job also specify SHR. If you code DISP=SHR, the system assumes DISP=(SHR,PASS).

3. DISP=(NEW,PASS)

   The library is created and used in the job, and is deleted at the end of the job.

4. DISP=(NEW,CATLG)

   The library is created, cataloged, and used in the job, and is kept at the end of the job.

When the Library Is Cataloged

If the private library is cataloged, you must always code the DSNAME and DISP parameters.

- The DSNAME parameter specifies the name of the private library.
- The DISP parameter is either DISP=(OLD,PASS) or DISP=(SHR,PASS).

The other parameter you might code is DCB.

- Code the DCB parameter if complete data control block information is not contained in the data set label.

If you wish to refer to the private library in a later DD statement, code DSNAME=*.JOBLIB and the DISP parameter, DISP=(OLD,disposition). (Do not assign a disposition of DELETE, because the library would then be deleted at the end of the job step and be unavailable for use during the remainder of the job.) If a later DD statement defines a data set that is to be placed on the same volume as the private library, you can code VOLUME=REF=*.JOBLIB to obtain volume and unit information.

When the Library is Not Cataloged

If the private library is not cataloged, you must always code the DISP and UNIT parameters.

- The DISP parameter is

  DISP=(OLD,PASS),DISP=(SHR,PASS),DISP=(NEW,PASS), or DISP=(NEW,CATLG).

- The UNIT parameter specifies the device to be allocated to the library.

You must always code the VOLUME parameter unless the status of the data set is NEW. The DSNAME parameter is required unless the data set has been assigned a disposition of (NEW,PASS). If the status of the data set is NEW, the SPACE parameter is required.

- The VOLUME parameter identifies the volume serial number.
- The DSNAME parameter specifies the name of the private library.
- The SPACE parameter allocates space for the library on the designated volume.

The other parameter you might code is DCB.

- Code the DCB parameter if complete data control block information is not contained in the data set label.

If you wish to refer to the private library in a later DD statement, code DSNAME=*.JOBLIB, VOLUME=REF=*.JOBLIB (or VOLUME=SER=serial number, UNIT=unit information), and the DISP parameter, DISP=(OLD,disposition). (Do not assign a disposition of DELETE, because the library would then be deleted at the end of the job step and be unavailable for use during the remainder of the job). If a later DD statement defines a data set that is to be placed on the same volume as the private library, you can code VOLUME=REF=*.JOBLIB to obtain volume and unit information.

Concatenating Libraries

You can arrange a sequence of DD statements that define different libraries. The libraries are searched in the order in which the DD statements appear. If the system library is not defined on one of these DD statements, it is searched last.

To concatenate libraries, omit the ddname from all the DD statements defining the libraries except the first DD statement. The first DD statement must specify a ddname of JOBLIB, and the entire group must appear immediately after the JOB statement.

When the Job Includes a STEPLIB DD Statement

If both JOBLIB and STEPLIB DD statements appear in a job, the STEPLIB definition has precedence, i.e., the private library defined by the JOBLIB DD statement is not searched for any step that contains the STEPLIB definition. If you want the JOBLIB definition ignored but the step does not require use of another private library, define the system library on the STEPLIB DD statement:

```
//STEPLIB DD DSNAME=SYS1.LINKLIB,DISP=SHR
```

Examples of the JOBLIB DD Statement

1.
```
//PAYROLL JOB
//JOBLIB  DD   DSNAME=PRIVATE.LIB4,DISP=(OLD,PASS)
//STEP1   EXEC PGM=SCAN
//STEP2   EXEC PGM=UPDATE
//DD1     DD   DSNAME=*.JOBLIB,DISP=(OLD,PASS)
```

The private library defined on the JOBLIB DD statement is cataloged. The statement named DD1 refers to the private library defined in the JOBLIB DD statement.

2.
```
//PAYROLL JOB  REGION=86K
//JOBLIB  DD   DSNAME=PRIV.DEPT58,DISP=(OLD,PASS),UNIT=2311,        X
//            VOLUME=SER=D58PVL
//STEP1   EXEC PGM=DAY
//STEP2   EXEC PGM=BENEFITS
//DD1     DD   DSNAME=*.JOBLIB,VOLUME=REF=*.JOBLIB,DISP=(OLD,PASS)
```

The private library defined on the JOBLIB DD statement is not cataloged. The statement named DD1 refers to the private library defined in the JOBLIB DD statement.

3.
```
//TYPE    JOB  MSGLEVEL=(1,1)
//JOBLIB  DD   DSNAME=GROUP8.LEVEL5,DISP=(NEW,CATLG),UNIT=2311,     X
//            VOLUME=SER=148562,SPACE=(CYL,(50,3,4))
//STEP1   EXEC PGM=DISC
//DDA     DD   DSNAME=GROUP8.LEVEL5(RATE),DISP=OLD,                 X
//            VOL=REF=*.JOBLIB
//STEP2   EXEC PGM=RATE
```

The private library defined on the JOBLIB DD statement does not exist yet; therefore, all the parameters required to define the private library are included on the JOBLIB DD statement. The library is not created until STEP1 when a new member is defined for the library. The system looks for the program named DISC in the system library; the system looks for the program named RATE first in the private library.

4.
```
//PAYROLL JOB
//JOBLIB  DD   DSNAME=KRG.LIB12,DISP=(OLD,PASS)
//        DD   DSNAME=GROUP31.TEST,DISP=(OLD,PASS)
//        DD   DSNAME=PGMSLIB,UNIT=2311,                            X
//             DISP=(OLD,PASS),VOLUME=SER=34568
```

Several private libraries are concatenated. The system searches for each program in this order: KRG.LIB12, GROUP31.TEST, PGMSLIB, before searching SYS1.LINKLIB.

STEPLIB

Unless the system is told that the program requested on the EXEC statement resides in a private or temporary library, the system expects to find the program in the system library (SYS1.LINKLIB). One way to tell the system that the program the job step needs resides in a private library is to include, as one of the DD statements for that step, a DD statement named STEPLIB. (The other way to tell the system that a program resides in a private library is to follow the JOB statement with a DD statement named JOBLIB. The JOBLIB DD statement is described in the previous topic, "JOBLIB.") If you include a STEPLIB DD statement, each time a program is requested the system first looks in the private library for the program the job step uses; if the system does not find the program there, it looks for the program in the system library.

RULES FOR CODING THE STEPLIB DD STATEMENT

1. The ddname must be STEPLIB. Never use the ddname STEPLIB except when you are defining a private library.

2. A STEPLIB DD statement can appear in any position among the DD statements for the step.

3. The library defined on a STEPLIB DD statement can be referred to by or passed to later job steps in the same job.

4. A STEPLIB DD statement can appear in a cataloged procedure.

5. The parameters you code on the STEPLIB DD statement are determined by whether the library is cataloged, not cataloged, or passed by a previous job step.

When the Library is Cataloged

If the private library is cataloged, you must always code the DSNAME and DISP parameters.

- The DSNAME parameter specifies the name of the private library.
- The DISP parameter specifies the library's status, either OLD or SHR, and its disposition. The disposition would be KEEP, UNCATLG, DELETE, or PASS, depending on how you want the library treated after its use in the job step.

The other parameter you might code is DCB.

- Code the DCB parameter if complete data control block information is not contained in the data set label.

When the Library Is Not Cataloged or Passed

If the private library is not cataloged or passed, you must always code the DSNAME, DISP, VOLUME, and UNIT parameters.

- The DSNAME parameter specifies the name of the private library.
- The DISP parameter specifies the library's status, either OLD or SHR, and its disposition. The disposition would be KEEP, CATLG, DELETE, or PASS, depending on how you want the library treated after its use in the job step.
- The VOLUME parameter identifies the volume serial number.
- The UNIT parameter specifies the device to be allocated to the library.

The other parameter you might code is DCB.

- Code the DCB parameter if complete data control block information is not contained in the data set label.

When the Library Is Passed By a Previous Step

If a private library has been assigned a disposition of PASS, a later job step can use the library when you code the DSNAME and DISP parameters on a STEPLIB DD statement.

- The DSNAME parameter specifies either the name of the private library or a backward reference of the form *.stepname.STEPLIB. If the STEPLIB DD statement that assigned a disposition of PASS occurs in a cataloged procedure, the backward reference must include the procedure step name, i.e., *.stepname.procstepname.STEPLIB.
- The DISP parameter specifies a status of OLD and a disposition. The disposition would be KEEP, CATLG, UNCATLG, DELETE, or PASS, depending on how you want the library treated after its use in the job step.

Concatenating Libraries

You can arrange a sequence of DD statements that define different libraries. The libraries are searched in the order in which the DD statements appear. If the system library is not defined on one of these statements, it will be searched last for the program the job step uses.

To concatenate libraries, omit the ddname from all the DD statements defining the libraries except the first DD statement. The first DD statement must specify a ddname of STEPLIB, and the entire group appears as part of the DD statements for a particular step.

When the Job Includes a JOBLIB DD Statement

If both JOBLIB and STEPLIB DD statements appear in a job, the STEPLIB definition has precedence, i.e., the private library defined by the JOBLIB DD statement is not searched for any step that contains the STEPLIB definition. If you want the JOBLIB definition ignored but the step does not require use of another private library, define the system library on the STEPLIB DD statement:

```
//STEPLIB  DD  DSNAME=SYS1.LINKLIB,DISP=OLD
```


Examples of the STEPLIB DD Statement

1.
```
//PAYROLL JOB
//STEP1   EXEC LAB14
//STEP2   EXEC PGM=SPKCH
//STEPLIB DD   DSNAME=PRIV.LIB5,DISP=(OLD,KEEP)
//STEP3   EXEC PGM=TIL80
//STEPLIB DD   DSNAME=PRIV.LIB13,DISP=(OLD,KEEP)
```

The private libraries defined in STEP2 and STEP3 are cataloged.

2.
```
//PAYROLL JOB
//JOBLIB  DD   DSNAME=LIB5.GROUP4,DISP=(OLD,PASS)
//STEP1   EXEC PROC=SNZ12
//STEP2   EXEC PGM=SNAP10
//STEPLIB DD   DSNAME=LIBRARYP,DISP=(OLD,PASS),                    X
//             UNIT=2311,VOLUME=SER=55566
//STEP3   EXEC PGM=A1530
//STEP4   EXEC PGM=SNAP11
//STEPLIB DD   DSNAME=*.STEP2.STEPLIB,                             X
//             DISP=(OLD,KEEP)
```

The private library defined in STEP2 is not cataloged. The STEPLIB DD statement in STEP4 refers to the library defined in STEP2. Since a JOBLIB DD statement is included, STEP1 and STEP3 could execute programs from LIB5.GROUP4 or, if not found there, from SYS1.LINKLIB. STEP2 and STEP4 could execute programs from LIBRARYP or SYS1.LINKLIB.

3.
```
//PAYROLL JOB
//JOBLIB  DD   DSNAME=LIB5.GROUP4,DISP=(OLD,PASS)
//STEP1   EXEC PGM=SUM
//STEPLIB DD   DSNAME=SYS1.LINKLIB,DISP=OLD
//STEP2   EXEC PGM=VARY
//STEP3   EXEC PGM=CALC
//STEPLIB DD   DSNAME=PRIV.WORK,DISP=(OLD,PASS)
//        DD   DSNAME=LIBRARYA,DISP=(OLD,KEEP),                    X
//             UNIT=2311,VOLUME=SER=44455
//        DD   DSNAME=LIB.DEPT88,DISP=(OLD,KEEP)
//STEP4   EXEC PGM=SHORE
```

STEP2 and STEP4 can use programs contained in the private library named LIB5.GROUP4, which is defined in the JOBLIB DD statement. STEP1 can use a program only from the system library, since the library defined on the STEPLIB DD statement is the system library and the JOBLIB definition is ignored. A concatenation of private libraries is defined in STEP3. The system searches for the program named CALC in this order: PRIV.WORK, LIBRARYA, LIB.DEPT88, SYS1.LINKLIB. If a later job step refers to the STEPLIB DD statement in STEP3, the system will search for the program in the private library named PRIV.WORK, and if not found there, in SYS1.LINKLIB.

SYSABEND and SYSUDUMP

Each job step may contain one DD statement with a ddname of either SYSABEND or SYSUDUMP; if both are included, the last statement is used. These DD statements define a data set in which an abnormal termination dump can be written if the job step abnormally terminates. (Never use the ddname SYSABEND or SYSUDUMP unless you are defining a data set in which a dump can be written.) The dump provided when the SYSABEND DD statement is used includes the system nucleus, the processing program storage area, and a trace table, if the trace table option (MFT only) was requested at system generation. The SYSUDUMP DD statement provides only a dump of the processing program storage area.

The parameters you code on one of these statements are determined by whether you want the dump written to a unit record device or stored and written at a later time.

WRITING THE DUMP TO A UNIT RECORD DEVICE

If you want the dump written to a unit record device, you code either the UNIT or SYSOUT parameter.

- The UNIT parameter specifies the unit record device to which you want to write the dump, e.g., UNIT=1403.
- The SYSOUT parameter specifies the output class through which you want the data set routed, e.g., SYSOUT=A.

If the SYSOUT parameter is coded, the dump is not routed directly to a system output device. Instead, the dump is stored on a direct access device and later written on a system output device. If you want control over which direct access device the dump is stored on, you can include the UNIT parameter. You can also control the amount of space allocated to the dump by including the SPACE parameter. Otherwise, the system assigns a direct access device and space for a dump. (The device and space that the system assigns are specified as PARM parameter fields in the cataloged procedure for the input reader.) If you require a great deal of space for dumping, you should request enough space with the SPACE parameter rather than using the default; if there is not enough space, the system will not write the dump and will not issue a diagnostic message.

STORING THE DUMP

If you want to store the dump and write it at a later time, the DD statement must include the DSNAME, UNIT, VOLUME, and DISP parameters.

- The DSNAME parameter specifies the name of the data set.
- The UNIT parameter specifies the device to allocate to the data set.

- The VOLUME parameter identifies the volume serial number.

- The DISP parameter specifies the data set's status and disposition. Since you want to store the data set, the data set's disposition must be either KEEP, CATLG, or PASS.

If the dump is to be stored on a direct access device, you must code either the SPACE, SPLIT, or SUBALLOC parameter.

- The SPACE, SPLIT, or SUBALLOC parameter specifies the amount of space you want allocated to the data set. (Note: Excluding the requirements of the BSAM modules, the control program requires 2784 bytes of main storage within the partition of the failing task to provide dumps. Of the 2784 bytes 1344 are required for EOV processing should the initial space specifications for a direct access device be exceeded. This should be taken into consideration when making the specification on the SYSABEND or SYSUDUMP DD statements.)

Reference:

Refer to the Programmer's Guide to Debugging for information on how to interpret dumps.

Examples of the SYSABEND and SYSUDUMP DD Statements

1.
```
//STEP2       EXEC PGM=A
//SYSABEND    DD   SYSOUT=A
```

The SYSABEND DD statement specifies that you want the dump routed through the standard output class A.

2.
```
//STEP3       EXEC PGM=B
//SYSUDUMP    DD   SYSOUT=F,SPACE=(TRK,(0,50)),UNIT=(2311,3)
```

The SYSUDUMP DD statement specifies that you want the dump routed through the output class F. The dump is temporarily stored on the specified device. If the UNIT and SPACE parameters were not coded, the system would assign a direct access device and an estimate of space required for the dump. In the SPACE parameter, zero tracks are requested for the primary quantity; therefore, no space is allocated unless the step abnormally terminates. If the step abnormally terminates, space for a dump is allocated using the secondary quantity. Requesting multiple units increases the likelihood that one of the volumes mounted on these devices contains enough space to allocate the secondary quantity.

3.
```
//STEP1       EXEC PGM=PROGRAM1
//SYSABEND    DD   DSNAME=DUMP,UNIT=2311,DISP=(,PASS,KEEP),        X
//                 VOLUME=SER=1234,SPACE=(TRK,(110,10))
//STEP2       EXEC PGM=PROGRAM2
//SYSABEND    DD   DSNAME=*.STEP1.SYSABEND,DISP=(OLD,DELETE,KEEP)
```

The SYSABEND DD statements specify that you want the dump stored. The space request in STEP1 is large (110 tracks) so that the dumping operation is not inhibited due to insufficient space; if STEP1 does not abnormally terminate but STEP2 does, the dump will be written using the space allocated in STEP1. In both steps, a conditional disposition of KEEP is specified. This allows storing of the dump if either of the steps abnormally terminates. If both of the steps are successfully executed, the second term of the DISP parameter (DELETE) in STEP2 causes the data set to be deleted and the space acquired for dumping to be freed.

```
4.  //STEP1        EXEC PGM=WWK
    //SYSUDUMP     DD   DSNAME=DUMP,UNIT=2311,DISP=(,DELETE,          X
    //                  KEEP),VOLUME=SER=54366,SPACE=(TRK,(80,10))
    //STEP2        EXEC PGM=PRINT,COND=ONLY
    //IN           DD   DSNAME=*.STEP1.SYSUDUMP,DISP=(OLD,DELETE),   X
    //                  VOLUME=REF=*.STEP1.SYSUDUMP
```

STEP1 specifies that the dump is to be stored if the step abnormally terminates. Because COND=ONLY is specified in STEP2, the step is executed only if STEP1 abnormally terminates. STEP2 uses a program that prints the dump.

SYSCHK

If CHKPT macro instructions were executed during the original execution of your processing program, checkpoint entries were written on a checkpoint data set. If you plan to resubmit your job for restart and execution is to be restarted at a particular checkpoint, you must include a DD statement named SYSCHK when you resubmit the job. The SYSCHK DD statement defines the data set on which the checkpoint entry was written.

RULES FOR CODING THE SYSCHK DD STATEMENT

1. The ddname must be SYSCHK. SYSCHK can be used as the ddname of other DD statements in jobs.

2. The SYSCHK DD statement must immediately precede the first EXEC statement of the resubmitted job when restart is to begin at a checkpoint. (If the first EXEC statement is preceded by a DD statement named SYSCHK and restart is to begin at a step, the SYSCHK DD statement is ignored.)

3. If a JOBLIB DD statement is included, the SYSCHK DD statement must follow it.

4. The RESTART parameter must be coded on the JOB statement; otherwise, the SYSCHK DD statement is ignored.

5. The parameters you code on the SYSCHK DD statement are determined by whether the checkpoint data set is cataloged.

When the Checkpoint Data Set Is Cataloged

If the checkpoint data set is cataloged, you must always code the DSNAME and DISP parameters.

- The DSNAME parameter specifies the name of the checkpoint data set.

- The DISP parameter must specify or imply a status OLD and a disposition of KEEP.

Other parameters you might code are VOLUME, UNIT, LABEL, and DCB.

- If the checkpoint entry exists on a tape volume other than the first volume of the checkpoint data set, you must indicate this by coding the volume serial number or volume sequence number in the VOLUME parameter. (The serial number of the volume on which a checkpoint entry was written is contained in the console message printed after the checkpoint entry is written.) If you code the volume serial number, you must also code the UNIT parameter, since the system will not look in the catalog for unit information.

- Code the LABEL parameter if the checkpoint data set does not have standard labels.
- Code DCB=TRTCH=C if the checkpoint data set is on 7-track magnetic tape with nonstandard labels or no labels.

When the Checkpoint Data Set Is Not Cataloged

If the checkpoint data set is not cataloged, you must always code the DSNAME, DISP, VOLUME, and UNIT parameters.

- The DSNAME parameter specifies the name of the checkpoint data set. If the checkpoint data set is partitioned, do not include a member name in the DSNAME parameter.
- The DISP parameter must specify or imply a status of OLD and disposition of KEEP.
- The VOLUME parameter specifies the volume serial number of the volume on which the checkpoint entry resides. (The serial number of the volume on which a checkpoint entry was written is contained in the console message printed after the checkpoint entry is written.)
- The UNIT parameter specifies the device to be allocated to the data set.

Other parameters you might code are LABEL and DCB.

- Code the LABEL parameter if the checkpoint data set does not have standard labels.
- Code DCB=TRTCH=C if the checkpoint data set is on 7-track magnetic tape with nonstandard or no labels.

Examples of the SYSCHK DD Statement

1.
```
//JOB1    JOB  RESTART=(STEP3,CK3)
//SYSCHK  DD   DSNAME=CHLIB,UNIT=2311,                          X
//            DISP=OLD,VOLUME=SER=456789
//STEP1   EXEC
```

The checkpoint data set defined on the SYSCHK DD statement is not cataloged.

2.
```
//JOB2    JOB  RESTART=(STEP2,NOTE2)
//JOBLIB  DD   DSNAME=PRIV.LIB3,DISP=(OLD,PASS)
//SYSCHK  DD   DSNAME=CHECKPTS,DISP=(OLD,KEEP),                 X
//            UNIT=2400,VOLUME=SER=438291
//STEP1   EXEC
```

The checkpoint data set defined on the SYSCHK DD statement is not cataloged. Note that the SYSCHK DD statement follows the JOBLIB DD statement.

3.
```
//JOB3    JOB  RESTART=(*,CHECK4)
//SYSCHK  DD   DSNAME=CHKPTLIB,DISP=OLD,                        X
//            LABEL=(,NSL),DCB=(TRTCH=C)
//STEP1   EXEC
```

The checkpoint data set defined on the SYSCHK DD statement is cataloged and has nonstandard labels.

The * Parameter

```
//ddname  DD   *
```

*

specifies that the data following this statement is to be entered through the input stream for use by a processing program.

Rules for Coding

1. You may code more than one DD * statement per job step.
2. When you call a cataloged or in-stream procedure, you may add more than one DD * statement to a procedure step.
3. If the data is preceded by a DD * statement, a delimiter statement following the data is optional.
4. Only the keywords DLM and DIAGNS and the DCB subparameters BLKSIZE and BUFNO can be coded on the DD * statement. Any other parameters coded on a DD * statement are flagged as errors and the job fails.
5. A cataloged or in-stream procedure cannot contain a DD * statement.
6. Code the DATA parameter instead of the * parameter when the data contains job control statements.

Defining Data in the Input Stream

The input stream can be on a card reader, a magnetic tape, or a direct access device.

If the EXEC statement for the job step specifies a program name, you can include the data for the job step in the input stream. If the EXEC statement for the job step calls a cataloged or in-stream procedure, you can include the data for each procedure step in the input stream.

If the processing program does not read all the data in an input stream, the remaining data is flushed without causing abnormal termination of the job.

You can include several distinct groups of data in the input stream for a job step or procedure step. The system will recognize each group of data if you precede each group with a DD * statement, or follow each group with a delimiter statement, or both. If you code the DLM parameter on the DD * statement, you must terminate the data with the value assigned in the DLM parameter. If you do not code the DLM parameter, the delimiter is /* and is not required. If you leave out the DD * statement for a group of data, the system provides a DD * statement having SYSIN as its ddname.

The following rules apply when data is entered through an input stream:

- The input stream can be on any device supported by QSAM.
- The characters in the records must be coded in BCD or EBCDIC.

If you require some other mode of processing, such as column binary mode (DCB=MODE=C), arrange with your installation to provide your program

with a card reader at the time your job step executes. Then request the use of the card reader by a DD statement specifying UNIT and DCB information and provide the operators with the cards to be read under the special processing mode.

Note: When the automatic SYSIN batching reader is used to read the input stream, a DD * statement does not appear in the output listing. Instead, an identically named DD statement describing the temporary data set created from the input data appears.

The DCB Subparameters BLKSIZE, BUFNO, and DIAGNS

The input reader procedure causes data in the input stream to be written onto a direct access device so that the data can be retrieved rapidly when it is required by a processing program. As the data is written onto the direct access device, the data may be blocked. The block size and number of buffers used for blocking the data is established in the input reader procedure assigned to read the input stream. If you want shorter blocks than would be the case if the block size in the input reader procedure were assumed, you can specify the desired block size. (You cannot request larger blocks.)

To specify the desired block size, code DCB=BLKSIZE=blocksize on the DD * statement. To decrease the number of buffers, include the DCB subparameter BUFNO, e.g., DCB=(BLKSIZE=80,BUFNO=1). (When a job is submitted via remote job entry and the DCB subparameter BUFNO is coded on a DD * statement, BUFNO is ignored.)

BLKSIZE, BUFNO, and DIAGNS may be coded on a DD statement that contains the DDNAME parameter, which refers to another DD statement. (You cannot use a backward reference to a previously-defined DD statement to obtain these DCB subparameters; they must be coded explicitly on the DD statement that contains the DDNAME parameter.) If, in turn, the referenced DD statement defines data in the input stream, the BLKSIZE and BUFNO DCB subparameters are used to block the data. However, if the referenced DD statement contains its own DCB subparameters BLKSIZE and BUFNO, these values override those on the DD statement that contains the DDNAME parameter.

Examples of the * Parameter

1.
```
//INPUT1      DD   *
              .
              .
            data
              .
              .
/*
//INPUT2      DD   *,DLM=HD
              .
              .
            data
              .
              .
HD
```

The DLM parameter on the DD statement named INPUT2 defines a delimiter HD. This delimiter is used in place of /* to terminate the data defined in the input stream by INPUT2.

2.
```
//STEP2         EXEC       PROC=FRESH
//SETUP.WORK    DD         UNIT=2400,LABEL=(,NSL)
//SETUP.INPUT1  DD         *
                 .
                 .
               data
                 .
                 .
/*
//PRINT.FRM          DD    UNIT=180
//PRINT.INP          DD    *
                 .
                 .
               data
                 .
                 .
/*
```

The input data defined by the DD statement named SETUP.INPUT1 is for use by the cataloged procedure step named SETUP; the input defined by the DD statement named PRINT.INP is for use by the cataloged procedure step named PRINT.

3.
```
//INPUT2             DD    *,DCB=(BLKSIZE=1600,BUFNO=2,DIAGNS=TRACE)
                 .
                 .
               data
                 .
                 .
/*
```

The DCB subparameters BLKSIZE and BUFNO override those specified in the input reader procedure. The DCB subparameter DIAGNS requests the OPEN/CLOSE/EOV trace option.

The DATA Parameter

```
//ddname  DD  DATA
```

DATA
specifies that the data following this statement is to be entered through the input stream for use by a processing program. This data contains job control statements (i.e., these statements have the characters // in columns 1 and 2.)

Rules for Coding

1. The data cannot contain statements with /* (or the delimiter you assigned in the DLM parameter) in columns 1 and 2.

2. You may code more than one DD DATA statement per job step.

3. When you call a cataloged or in-stream procedure, you may add more than one DD DATA statement to a procedure step.

4. Each group of data must be preceded by a DD DATA statement and followed by a delimiter statement (/*).

5. Only the keywords DLM and DIAGNS and the DCB subparameters BLKSIZE and BUFNO can be coded on the DD DATA statement. Any other parameters coded on a DD DATA statement are flagged as errors and the job fails.

6. A cataloged or in-stream procedure cannot contain a DD DATA statement.

7. The * parameter may be coded instead of the DATA parameter when the data does not contain job control statements.

Defining Data in the Input Stream

The input stream can be on a card reader, a magnetic tape, or a direct access device.

If the EXEC statement for the job step specifies a program name, you can include the data for the job step in the input stream. If the EXEC statement for the job step calls a cataloged or in-stream procedure, you can include the data for each procedure step in the input stream.

If the processing program does not read all the data in an input stream, the remaining data is flushed without causing abnormal termination of the job.

You can include several distinct groups of data in the input stream for a job step or procedure step. The system will recognize each group of data if you precede each group with a DD DATA statement and follow each group with a delimiter statement. If you code the DLM parameter on the DD DATA statement, the delimiter terminating the group of data is the value assigned in the DLM parameter. Otherwise, the delimiter is /*.

The following rules apply when data is entered through an input stream:

- The input stream can be on any device supported by QSAM.
- The characters in the records must be coded in BCD or EBCDIC.

If you require some other mode of processing, such as column binary mode (DCB=MODE=C), arrange with your installation to provide your program with a card reader at the time your job step executes. Then request the use of the card reader by a DD statement specifying UNIT and DCB information:

```
//name     DD    UNIT=2540-R,DCB=(MODE=C,BLKSIZE=160)
```

and provide the operations staff with the deck of cards to be read under the special processing mode.

Note: When the automatic SYSIN batching reader is used to read the input stream, a DD DATA statement does not appear in the output listing. Instead, an identically named DD statement describing the temporary data set created from the input data appears.

The DCB Subparameters BLKSIZE, BUFNO, and DIAGNS

The input reader procedure causes data in the input stream to be written onto a direct access device so that the data can be retrieved rapidly when it is required by a processing program. As the data is written onto the direct access device, the data may be blocked. The block size and number of buffers used for blocking the data is established in the input reader procedure assigned to read the input stream. If you want shorter blocks than would be the case if the block size in the input reader procedure were assumed, you can specify the desired block size. (You cannot request larger blocks.)

To specify the desired block size, code DCB=BLKSIZE=blocksize on the DD DATA statement. To decrease the number of buffers, include the DCB subparameter BUFNO, e.g., DCB=(BLKSIZE=80,BUFNO=1). (When a job is submitted via remote job entry and the DCB subparameter BUFNO is coded on a DD DATA statement, BUFNO is ignored.)

BLKSIZE, BUFNO, and DIAGNS may be coded on a DD statement that contains the DDNAME parameter, which refers to another DD statement. If, in turn, the referenced DD statement defines data in the input stream, the BLKSIZE and BUFNO DCB subparameters are used to block the data. However, if the referenced DD statement contains its own DCB subparameters BLKSIZE and BUFNO, these values override those on the DD statement that contains the DDNAME parameter.

The ASB reader can not process the DDNAME parameter for in-stream procedures. For ASB in-stream procedures, code your DCB subparameter on the DD * or DD DATA card if you want shorter blocks than the assured blocksize in the ASB procedure.

Examples of the DATA Parameter

1.
```
//INPUT1  DD   DATA,DLM=AD,DCB=DIAGNS=TRACE
          .
          .
          data
          .
          .
AD
```

Defining data in the input stream. The DLM parameter defines a delimiter, AD. This delimiter is used in place of /* to terminate the data defined in the input stream. The DCB subparameter DIAGNS requests the OPEN/CLOSE/EOV trace option.

2.
```
//STEP2         EXEC PROC=UPDATE
//PREP.DD4      DD   DSNAME=A.B.C,VOLUME=SER=D88,UNIT=2311,        X
//                   SPACE=(TRK,(10,5)),DISP=(,CATLG,DELETE)
//PREP.INPUT    DD   DATA
                .
                .
              data
                .
                .
/*
//ADD.DD6       DD   SPACE=(TRK,(5,1))
//ADD.IN        DD   *
                .
                .
              data
                .
                .
/*
```


Defining data in the input stream. The input defined by the DD statement named PREP.INPUT is for use by the cataloged procedure step named PREP. This data contains job control statements. The input defined by the DD statement named ADD.IN is for use by the cataloged procedure step named ADD. Since this data is defined by a DD * statement, it must not contain job control statements.

3.
```
//INPUT2  DD   DATA,DCB=(BLKSIZE=400,BUFNO=1)
          .
          .
        data
          .
          .
/*
//INPUT3  DD   DATA
          .
          .
        data
          .
          .
/*
```

Defining several groups of data in the input stream. The DCB subparameters coded on the DD statement named INPUT2 are used to block the data that follows that statement.

The DUMMY Parameter

```
//ddname  DD   DUMMY
```

DUMMY
specifies that no device or external storage space is to be allocated to the data set, no disposition processing is to be performed on the data set, and, for BSAM and QSAM, specifies that no input or output operations are to be performed on the data set.

Rules for Coding

1. You can code the DUMMY parameter by itself or follow it with all of the parameters, except the DDNAME parameter, necessary to define a data set. DUMMY and DDNAME are mutually exclusive.

2. If the DUMMY parameter is coded and an access method other than the basic sequential access method (BSAM) or queued sequential access method (QSAM) is requested to read or write the data set, or if the DUMMY parameter is coded and the access method of BDAM load mode (BSAM with DCB MACRF=WL) is requested, a programming error occurs.

3. Data sets concatenated to a DUMMY data set will also be treated as DUMMY data sets by the system.

What the DUMMY Parameter Does

When you use either the basic sequential or queued sequential access method, the DUMMY parameter allows your processing program to execute without performing input or output operations on a data set. When the processing program asks to write a dummy data set, the write request is recognized, but no data is transmitted. When the processing program asks to read a dummy data set, an end-of-data-set exit is taken immediately.

Besides bypassing input or output operations on a data set, the DUMMY parameter causes the UNIT, VOLUME, SPACE, and DISP parameters, when coded on the DD DUMMY statement, to be ignored (if coded, these parameters are checked for syntax). Therefore, no devices or external storage space is allocated to the data set and no disposition processing is performed on the data set.

If you know that certain parts of a program "work" and need not be processed each time the job is submitted for testing, the DUMMY parameter can help save time. The DUMMY parameter can also be used to suppress the writing of data sets, such as output listings, that you do not need.

Coding the DUMMY Parameter

You can code the DUMMY parameter by itself or follow it with all the parameters you would normally code when defining a data set. However, in one case you must code another parameter after the DUMMY parameter: when certain DCB information, not supplied in the DCB macro instruction, is required for the processing program to execute successfully. For example, when an OPEN routine requires a BLKSIZE specification to obtain buffers, and BLKSIZE is not specified in the DCB macro instruction, you should supply this information by coding the DCB parameter after the DUMMY parameter. When a DD statement that overrides a procedure DD statement contains the DUMMY parameter, the system will ignore all the parameters coded on the procedure DD statement except for the DCB parameter.

When you want input or output operations performed on the data set, replace the DD statement that contains the DUMMY parameter with a DD statement that contains all of the parameters required to define this data set. When a procedure DD statement contains the DUMMY parameter, you can nullify it by coding the DSNAME parameter on the overriding DD statement. However, be sure the data set name is not NULLFILE. Assigning the name NULLFILE in the DSNAME parameter has the same effect as coding DUMMY.

If you code DUMMY on a DD statement and a later DD statement in the same job refers to this DD statement when requesting unit affinity (UNIT=AFF=ddname) or volume affinity (VOLUME=REF=*.stepname.ddname), the data set defined on the later DD statement is assigned a dummy status.

Examples of the DUMMY Parameter

1.
```
//OUTPUT3      DD   DUMMY,DSNAME=X.Y.Z,UNIT=2311,                X
//                  SPACE=(TRK,(10,2)),DISP=(,CATLG)
```

   This DD statement defines a dummy data set. The parameters coded with the DUMMY parameter are not used.

2.
```
//IN       DD   DUMMY,DCB=(BLKSIZE=800,LRECL=400,RECFM=FB)
```

   This DD statement defines a dummy data set. The DCB parameter is coded to supply information for the data control block that was not supplied in the DCB macro instruction.

3. If you are calling a cataloged procedure that contains the following DD statement in STEP4

   ```
   //IN DD   DUMMY,DSNAME=ELLN,DISP=OLD,VOL=SER=11257,UNIT=2314
   ```

   you can nullify the effects of the DUMMY parameter by coding:

   ```
        //STEP4.IN  DD  DSNAME=ELLN
   ```

4. If you are calling a cataloged procedure that contains the following DD statement in STEP1

   ```
   //TAB  DD  DSNAME=APP.LEV12,DISP=OLD
   ```

   you can make this DD statement define a dummy data set by coding:

   ```
        //STEP1.TAB  DD  DUMMY
   ```

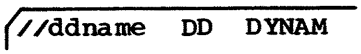
5. If you are calling a cataloged procedure that contains the following DD statement in a procedure step named LOCK

   ```
   //MSGS  DD  SYSOUT=A
   ```

   you can make this DD statement define a dummy data set by coding:

   ```
        //LOCK.MSGS  DD  DUMMY
   ```

The DYNAM Parameter



DYNAM
used in the TSO LOGON procedure to specify that dynamic allocation of data sets is to be used. This allows you to defer definition of a data set until you require it. If DYNAM is used in the background (batch environment), it means the same as DUMMY.

Rules for Coding

1. The dynamic allocation meaning of DYNAM is only effective for foreground jobs using an MVT system with TSO. For MFT, or MVT without TSO, DYNAM has the same meaning as coding DUMMY. Like DUMMY, DYNAM is a positional parameter.

2. No other parameters may be coded with the DYNAM parameter.

3. The DDNAME parameter cannot be used to refer to a DYNAM DD statement.

What the DYNAM Parameter Does

During LOGON processing for TSO, no devices or external storage are allocated to a data set defined by a DD DYNAM statement. The DYNAM parameter reserves space in internal tables so that data set requirements that arise during the terminal session may be satisfied. When you require a data set, the actual device and external storage for the data set can then be allocated.

When DYNAM is used in the background (batch environment) or in the foreground before allocation, it has the same effect as coding DUMMY. Refer to the section on the DUMMY parameter in this book for more information.

Coding the DYNAM Parameter

DYNAM is a positional parameter. However, no other parameters may be coded with DYNAM.

To nullify the DYNAM parameter in a cataloged procedure, code the SYSOUT or DSNAME parameter in the overriding DD statement, but do not use the DSNAME of NULLFILE.

Example of the DYNAM Parameter

1. `//INPUT DD DYNAM`

   For TSO, this statement specifies dynamic allocation may be requested. For background jobs, DYNAM has the same meaning as DUMMY.

The AFF Parameter

AFF=ddname

ddname
the name of an earlier DD statement in the same job step that requests processing of a data set on a separate channel from the one on which certain other data sets are being processed.

Rules for Coding

1. The DD statement that the AFF parameter refers to must contain the SEP parameter.

2. If channel separation is critical, use the UNIT parameter to specify a particular channel, using an absolute unit address or group name. (How to specify a particular channel is described in the chapter "The UNIT Parameter.")

3. The AFF, SEP, DDNAME, and SYSOUT parameters are mutually exclusive parameters; therefore, when SEP, DDNAME, or SYSOUT is coded, do not code the AFF parameter.

OPTIMIZING CHANNEL USAGE

The devices that the system allocates for data sets used in a job step are attached to channels. These channels transmit the data in the data sets from the device to the CPU. When two or more data sets are to be used in a job step, processing time may be shortened if the system transmits data over separate channels.

Requesting Channel Separation

The SEP and AFF parameters can be used to request channel separation. You list in the SEP parameter the names of up to eight earlier DD statements that define data sets from which channel separation is desired. (The SEP parameter is described in the chapter "The SEP Parameter" which appears later in this section.) Coding the AFF parameter is a shortcut method of requesting channel separation, since you list only one ddname and that ddname refers to an earlier DD statement in the same job step that contains the SEP parameter. The AFF parameter tells the system that you want the data set defined on this DD statement to have the same channel separation as the data set defined on the named DD statement. The AFF parameter does not tell the system that these two data sets are to be assigned to the same channel -- the system will decide that based on what devices are available for allocation.

If the system finds it impossible in the current environment to satisfy the channel separation request, the system may try to alter the current environment through some operator action. The operator is given the option of bringing a device online, cancelling the channel separation request, or cancelling the job. In certain environments, the operator may also be able to tell the system to wait for devices to become free. If you make a nonspecific request for a direct access volume and request channel separation, your request for separation may be ignored. This happens when the algorithm used to allocate data sets to devices is not able to select the device that would permit the desired channel separation.

Requests for channel separation are ignored for any data sets that have been allocated devices by the automatic volume recognition (AVR) option.

If it is essential that data be transmitted via a particular channel, you can specify an absolute unit address or group name (if the group of devices is associated with one channel) in the UNIT parameter.

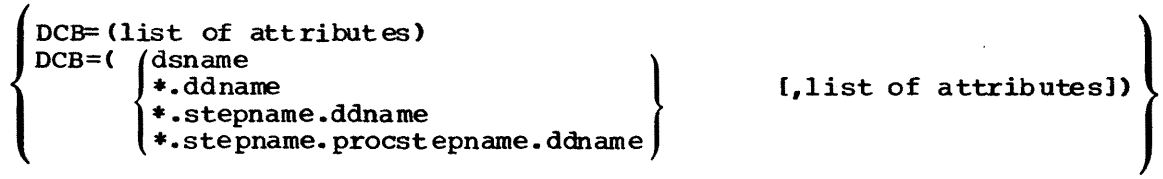
If neither the SEP nor AFF parameter is coded, any available channel, consistent with the UNIT parameter requirement, is assigned by the system.

Example of the AFF Parameter

1.
```
//STEP1    EXEC PGM=CONVERT
//INPUT1   DD   DSNAME=A.B.C,DISP=OLD
//INPUT2   DD   DSNAME=FILE,DISP=OLD,UNIT=2400,                    X
//             VOLUME=SER=54333
//BUF      DD   UNIT=2400,SEP=(INPUT1,INPUT2)
//OUTPUT   DD   DSNAME=ALPHA,UNIT=TAPE,DISP=(,KEEP),AFF=BUF
```

The system attempts to assign the data sets defined by the DD statements BUF and OUTPUT to a channel other than the ones assigned to the data sets defined by the DD statements INPUT1 and INPUT2. The data sets defined by the DD statements BUF and OUTPUT may or may not be assigned to the same channel. The parameter SEP=(INPUT1,INPUT2) could have been coded instead of AFF=BUF.

The DCB Parameter



list of attributes
those DCB keyword subparameters that describe the data set and are needed to complete the data control block. DCB keyword subparameters are listed in this chapter under "Glossary of DCB Subparameters."

dsname
specifies that the system is to copy DCB information from the data set label of a cataloged data set named "dsname." The cataloged data set must reside on a direct access volume and the volume must be mounted before execution of the job step. Specifying DCB=dsname is permitted only when DISP=OLD.

*.ddname
specifies that the system is to copy DCB information from an earlier DD statement in the same job step named "ddname."

*.stepname.ddname
specifies that the system is to copy DCB information from a DD statement named "ddname," which appears in an earlier job step named "stepname."

*.stepname.procstepname.ddname
specifies that the system is to copy DCB information from a DD statement named "ddname," which appears in a procedure step named "procstepname"; the procedure step is part of a cataloged or in-stream procedure that was called by an earlier jobstep named "stepname."

Rules for Coding

1. Separate each DCB keyword subparameter with a comma.

2. If the DCB parameter value consists of only one keyword subparameter, a data set name, or a backward reference, you need not enclose it in parentheses.

3. Only the DCB subparameters BLKSIZE, BUFNO and DIAGNS have meaning when coded with the DDNAME parameter. Do not code other DCB subparameters with the DDNAME parameter.

Completing the Data Control Block

Each data set that is to be read or written must have a data control block associated with it. The data control block is originally constructed in the processing program by a DCB macro instruction. This data control block can be completed when the DCB macro instruction is coded or at execution time through the DCB parameter on the DD statement and the data set label, if one exists.

When more than one source is used to complete the data control block, a merging process takes place (see Figure 9): first, information coded with the DCB macro instruction is placed in the data control block; then, information coded on the DD statement is placed in unfilled sections of the data control block; and, finally, information in the data set label, if one exists, is placed in still unfilled sections of the data control block. (DCB information may also be provided by default options assumed in the OPEN macro instruction and by your program, either before the data set is opened, by using the DCBD macro instruction, or in the DCB exit routine. Refer to the Data Management Macro Instructions publication.)

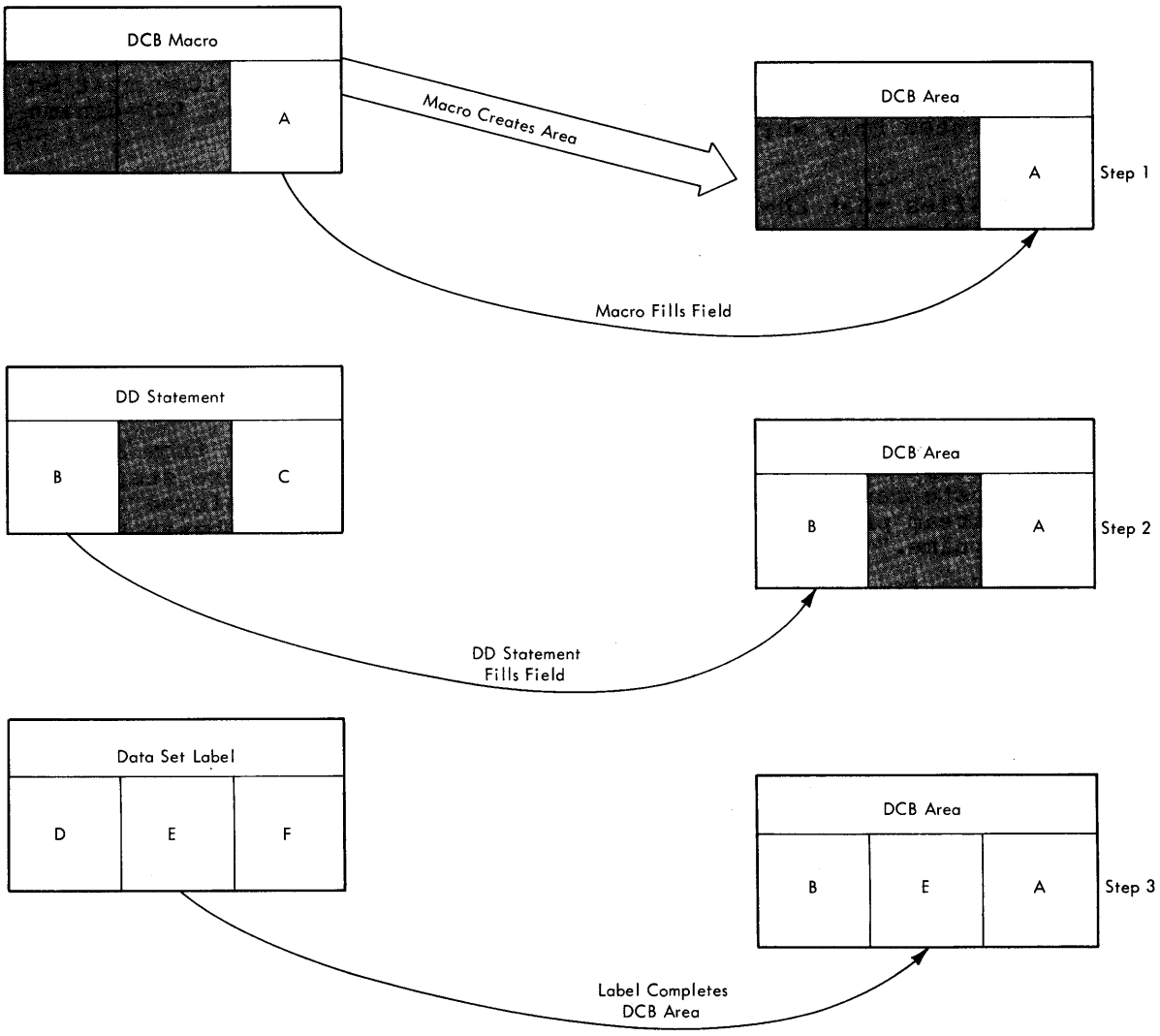


Figure 9. How the Data Control Block Is Filled

DCB Macro Instruction

The DCB macro instruction includes information about the data that is unlikely to change each time the processing program is executed. Also, it includes any information that is not related to the DCB parameter and the data set label (e.g., MACRF, DDNAME, EXLST).

DCB Parameter

The DCB parameter is coded on the DD statement and includes all the information that is not specified by any other source. How to specify DCB information on the DD statement is described in "Specifying DCB Information on the DD Statement."

Data Set Label

If the data set already exists and has standard labels, certain information is contained in the label that can be used to complete the data control block. For tape, the data set label can contain the data set's record format, block size, logical record length, tape recording density, and, for seven-track tape, tape recording technique. For direct access, the data set label can contain the data set's organization, record format, block size, logical record length, and if the data contains keys, the key length and relative key position.

Specifying DCB Information on the DD Statement

The DCB parameter must be coded on the DD statement unless the data control block is completed by other sources. There are several ways of specifying DCB information on the DD statement. You can:

- Supply all pertinent DCB keyword subparameters on the DD statement.
- Tell the system to copy DCB information from the data set label of an existing cataloged data set.
- Tell the system to copy DCB information from an earlier DD statement in the same job.

(Note: If a data set is old and data attributes are now specified which are variations of the first attributes, the old attributes will be ignored. You can override DCB attributes only on a DD statement contained in a cataloged or in-stream procedure, as discussed in Appendix A: Cataloged and In-Stream Procedures.)

SUPPLYING DCB KEYWORD SUBPARAMETERS

The DCB information required to complete the data control block can be listed as keyword subparameters in the DCB parameter; subparameters are separated by commas. If the processing program and the DCB parameter supply the same subparameter, the subparameter on the DD statement is ignored. Valid DCB keyword subparameters and the values that can be assigned to them are listed in this chapter under "Glossary of DCB Subparameters."

COPYING DCB INFORMATION FROM A DATA SET LABEL

To save time in coding the DCB parameter, you can tell the system to copy the DCB information from the data set label of a cataloged data set on a currently mounted direct access volume. The data set must have standard labels. A permanently resident volume is the most likely place from which to copy such information because it is always mounted. Code in the DCB parameter the data set name of the cataloged data set. The name you code cannot contain special characters, except for periods used in a qualified name.

The following DCB keyword subparameters can be copied from the data set label: DSORG, RECFM, OPTCD, BLKSIZE, LRECL, KEYLEN, and RKP. The volume sequence number and expiration date of the cataloged data set are also copied unless you specify these in the DD statement. If you code

any DCB keyword subparameters following the name of the cataloged data set, these subparameters override any of the corresponding subparameters that were copied. Valid DCB keyword subparameters and the values that can be assigned to them are listed in this chapter under "Glossary of DCB Subparameters."

COPYING DCB INFORMATION FROM AN EARLIER DD STATEMENT

Another way to save time in coding the DCB parameter is to tell the system to copy the DCB information from an earlier DD statement in the same job. The earlier DD statement can be contained in the same job step, an earlier job step or cataloged procedure step. If you code any DCB keyword subparameters following the reference to the DD statement, these subparameters override any of the corresponding subparameters that were copied. If the DD statement defines an existing data set and contains the DCB parameter, the system copies those subparameters from the earlier DD statement that were not previously specified for the existing data set. Valid DCB keyword subparameters and the values that can be assigned to them are listed below.

Glossary of DCB Subparameters

This glossary lists the keyword subparameters that you can code in the DCB parameter on a DD statement, their definitions, and the values that can be assigned to them. Across from each subparameter is a list of the access methods that use the subparameter.

Certain required subparameters cannot be coded in the DCB parameter, but must be coded in the DCB instruction. These subparameters are described in the Data Management Macro Instructions publication.

BFALN= {F / D} — Can be used with BDAM, BISAM, BPAM, BSAM, EXCP, QISAM, QSAM

Specifies the boundary alignment of each buffer as follows:

F -- each buffer starts on a fullword boundary that is not also a doubleword boundary.

D -- each buffer starts on a doubleword boundary.

If not specifed by any source, doubleword boundary alignment (D) is assumed. (Specifying BFALN=F in a DD statement causes fragmentation of main storage if the DCB is closed before releasing buffers.)

Note for QISAM: Buffer alignment information must be supplied from the same source as the type of buffering (BFTEK) information or both must be omitted.

BFTEK= {S / E / D / A / R} — Can be used with EXCP, QSAM, BTAM

Specifies the type of buffering to be used by the control program as follows:

For EXCP:

S -- simple buffering.

E -- exchange buffering.

For QSAM:

S -- simple buffering.

E -- exchange buffering (track overflow cannot be specified in the RECFM subparameter). Exchange buffering cannot be used with variable-length blocked or spanned records.

A -- record area buffering. In the locate mode with variable-length spanned records, the control program reads and writes entire logical records rather than segments.

If neither is specified by any source, simple buffering (S) is assumed.

Note for QSAM: The type of buffering information must be supplied from the same source as the boundary alignment (BFALN) information or both must be omitted.

For BTAM:

D -- dynamic buffering. If dynamic buffering is specified, a buffer pool must be defined.

For BDAM or BSAM:

R -- record buffering. For writing records in the create BDAM mode, this specification allows a logical record to span one or more tracks. For reading a data set, segments without keys are offset in the buffer by the key length. This means that the actual data starts in the same place in the buffer by the key length. This means that the actual data starts in the same place in the buffer for each read.

BLKSIZE=block size — Can be used with BDAM, BPAM, BSAM, QISAM (output only), QSAM, TCAN

For BDAM, BPAM, BSAM, QISAM, QSAM:

Specifies the maximum length, in bytes, of a block. The maximum length that can be specified is 32,760. For blocks of ASCII records on magnetic tape, the maximum length is 2048 and the minimum length is 18.

- If RECFM=F, then BLKSIZE must be ≥ logical record length.
- If RECFM=FB, then BLKSIZE must be an integral multiple of the logical record length.
- If RECFM=V, then BLKSIZE must be ≥ (logical record length + 4).
- If RECFM=VB, then BLKSIZE must be (n times logical record length) + 4; where n is the average number of logical records in the block rounded to the next highest number.

- If RECFM=D or DB, then BLKSIZE must be ≥ (maximum record length + block prefix length).

Note for QISAM: The block size that is specified must be at least 10 bytes less than the number of data bytes available on one track of the allocated direct access device. Key length can be included in the block size, but only as a part of the record length (LRECL). Do not separately add the key length to the block size. Block size information is necessary when creating a data set.

Note for BDAM, BPAM, BSAM, QSAM: If you code the BLKSIZE subparameter in the DCB macro instruction or on a DD statement that defines an existing data set and the data set has standard labels, the subparameter overrides the block size specified in the label.

Note for BDAM, BPAM, BSAM with keys: If direct-access device keys are used, do not include the key length in the block size. Specify key length in the KEYLEN subparameter of the DCB parameter.

Note for BSAM and QSAM with RECFM=FB: If the BLKSIZE subparameter on a DD statement for a SYSOUT data set (an output data set being routed through the output stream) is not an integral multiple of and larger than the logical record length (LRECL), the block size will be adjusted to the nearest lower multiple of the logical record length (LRECL).

Note for TCAM: TCAM specifies the length in bytes of the application program's work area into which TCAM will move message units to be processed. The number specified should be at least equal to the record length as specified by the LRECL operand and must not exceed 32,760. If OPTCD=W is specified, eight bytes must be included for the source of the message. If OPTCD=C is specified, one byte must be included to indicate the message segment. For variable length records, four bytes must be included for unblocked records or eight bytes for blocked records.

BUFIN=number of buffers Can be used with TCAM

Specifies the number of buffers to be assigned initially for receiving operations for each line in the line group. The number specified must be less than the number of buffers in the buffer pool for this line group and may not exceed 15. The number of buffers specified in the combined BUFIN and BUFOUT operands must be no greater than the number of buffers in the buffer pool for this line group (not including those for disk activity only). If This operand is omitted, 1 is assumed.

BUFL=buffer length Can be used with BDAM, BISAM, BPAM, BSAM, EXCP, QISAM, QSAM, TCAM

For BDAM, BISAM, BPAM, BSAM, EXCP, QISAM, QSAM:

Specifies the length, in bytes, of each buffer in the buffer pool. The maximum length is 32,760 bytes. Requirements for supplying buffer length information vary with the different data organization and access methods as follows:

BDAM -- required only if dynamic buffering is specified in the MACRF subparameter of the DCB macro instruction.

BPAM, BSAM, and QSAM -- optional. If omitted and the control program acquires buffers automatically, the sum of the block size (BLKSIZE) and key length (KEYLEN) is used to establish buffer length. If card image is specified (MODE=C), BUFL=160 must be specified.

BISAM and QISAM -- not required if the control program acquires buffers automatically or if dynamic buffering is specified. (For BISAM, dynamic buffering is specified in the MACRF subparameter of the DCB macro instruction).

Note: TCAM specifies the length in bytes of each of the Message Control Program buffers that handle messages received and sent by an application program. The length must be at least 31 bytes but may not exceed 65,535 bytes.

BUFMAX=number of buffers Can be used with TCAM

Specifies the maximum number of buffers to be allocated to a line at one time. The number specified must be greater than 1 but may not exceed 15 and must be at least equal to the larger of the numbers specified by BUFIN and BUFOUT. If this operand is omitted, 2 is assumed.

BUFNO=number of buffers Can be used with BDAM, BISAM, BPAM, BSAM, BTAM, EXCP, QISAM, QSAM

Specifies the number of buffers to be assigned to the data control block; the maximum number is 255, but the actual number allowed may be less than 255 because of limits established when the system was generated. Requirements for coding the BUFNO subparameter are as follows:

Method of Obtaining the Buffer Pool	Requirement for Indicating Number of Buffers
BUILD macro instruction (BDAM, BISAM, BPAM, QISAM, QSAM)	Must be specified.
GETPOOL macro instruction (BDAM, BISAM, BPAM, BSAM, QISAM, QSAM)	Control program uses the number specified in the GETPOOL macro instruction.
Automatically (BPAM and BSAM)	Must be specified.
Automatically (QISAM and QSAM)	Optional; if not specified, two buffers are obtained.
Dynamic buffering (BDAM and BISAM)	Optional; if not specified, two buffers are obtained.

BUFOFF= {n / L} Can be used with BSAM, QSAM

Specifies the buffer offset. The buffer offset is the length of an optional block prefix that may precede a block of one or more ASCII records on magnetic tape.

n -- the length of the block prefix. For input, n may be any unsigned decimal number from 0 through 99. For output, n can only be 0.

L -- the block prefix field is four bytes long and contains the block length. L may be specified only when record format (RECFM) is D.

BUFOUT=number of buffers Can be used with TCAM

Specifies the number of buffers to be assigned initially for sending operations for each line in the line group. The number specified must be less than the number of buffers in the buffer pool for this line group and may not exceed 15. The number of buffers specified in the combined BUFIN and BUFOUT operands must be no greater than the number of buffers in the buffer pool for this line group (not including those for disk activity only). If this operand is omitted, 2 is assumed.

BUFRQ=number of buffers Can be used with QTAM

Specifies the number of buffers to be requested in advance for the GET macro instruction. The maximum number is 255. If not specified by any source or if a value of less than 2 is specified, 2 is assumed. For information on calculating BUFRQ, refer to the publication IBM System/360 Operating System: Telecommunications Access Method Message Control, GC30-2005.

BUFSIZE=number Can be used with TCAM

Specifies the length in bytes of each of the buffers to be used for all lines in a particular line group. This length must be at least 31 bytes, but may not exceed 65,535. The buffer size should be an even multiple of the buffer-unit size as specified in the INTRO macro; the maximum number of buffer-units per buffer is 255.

CODE= {A, B, C, F, I, N, T} Can be used with BSAM, EXCP, QSAM

Specifies the paper tape code in which the data is punched.

A -- USASCII (8 track).
B -- Burroughs (7 track).
C -- National Cash Register (8 track).
F -- Friden (8 track).
I -- IBM BCD perforated tape and transmission code (8 track).
N -- No conversion required.
T -- Teletype (5 track).

If not specified by any source, I is assumed. The subparameters CODE, KEYLEN, MODE, PRTSP, STACK, and TRTCH are mutually exclusive subparameters. Therefore, if CODE is coded, do not code any of these other subparameters.

CPRI= {R, E, S} Can be used with QTAM

Specifies the relative priority to be given to sending and receiving operations, as follows:

R -- receiving has priority over sending. An output message is sent on a given line only during a polling interval.

E -- receiving and sending have equal priority. After each full polling sequence on a given line, all output messages queued for that line are transmitted.

S -- sending has priority over receiving. For nonswitched lines after QTAM polls a terminal on a line, the line is made available for outgoing messages, and the next terminal is polled only when there are no output messages in the queue for the line. For Auto Poll lines, the line is made available for outgoing messages after a message ending in EOT is received by a terminal on the line, or when the end of the polling list is reached. S must be specified for IBM 2740 Communications Terminals Types I and VI, and if the line group includes IBM 2740 Model 2 terminals.

If this subparameter is not specified by any source, CPRI=S is assumed.

This subparameter must be omitted if this line group consists of switched lines.

For WTTA Lines:

R or E -- output messages are sent when there is no traffic over the line, after an EOT character has been received, or after a time-out has occurred.

S -- output messages are sent when there is no traffic over the line, after an EOT or EOM character has been received, or after a time-out has occurred.

CYLOFL=number — Can be used with QISAM (output only)

Specifies the number of tracks on each cylinder to hold the records that overflow from other tracks on that cylinder. The maximum number is 99.

DEN= {0 / 1 / 2 / 3} — Can be used with BSAM, EXCP, QSAM

Specifies the magnetic tape density in number of bits-per-inch used to write a data set, as follows:

DEN	7-Track Tape	9-Track Tape	9-Track Tape (Phase Encoded)	9-Track Tape Dual Density
0	200			
1	556			
2	800	800		800(NRZI)
3			1600	1600(PE)

NRZI is for non-return-to-zero-inverse mode.
PE is for phase encoded mode.

Note: Specifying DEN=0 for a 7-track 3420 will result in a 556 bits-per-inch recording density, but corresponding messages and tape labels will indicate a 200 bits-per-inch recording density.

If not specified by any source, 800 bits-per-inch is assumed for 7-track tape, 800 bits-per-inch for 9-track without dual density, and 1600 bits-per-inch for 9-track with dual density or phase-encoded drives.

For 7-track tape, all information on the reel must be written in the same density (i.e., labels, data, tapemarks). Do not specify DEN for a SYSOUT data set.

DIAGNS=TRACE — Can be used with BDAM, BISAM, BPAM, BSAM, BTAM, EXCP, GAM, QSAM, QISAM, QTAM

Requests the Open, Close/EOV trace option which gives a module by module trace of Open/Close/EOV's workarea and the user's DCB.

If the subparameter is not specified on the DD card, the option is not implemented. The Generalized Trace Facility must be active in the system while the job that requested the trace is running.

DSORG=data set organization — Can be used with BDAM, BISAM, BPAM, BSAM, BTAM, EXCP, GAM, QISAM, QSAM, QTAM

Specifies the organization of the data set and whether the data set contains any location-dependent information that would make the data set unmovable (U). The values that can be used are as follows:

DA -- Direct access

DAU -- Direct access unmovable

CQ -- Direct access message queue or the checkpoint for a message control program. If this subparameter is not specified by any source, the telecommunications job, when executed, is terminated.

CX -- Communications line group

GS -- Graphic data control block

IS -- Indexed sequential

ISU -- Indexed sequential unmovable

MQ -- Data control block governing message transfer to or from a telecommunications message processing queue. If this subparameter is not specified by any source, the telecommunications job, when executed, is terminated.

PO -- Partitioned organization

POU -- Partitioned organization unmovable

PS -- Physical sequential

PSU -- Physical sequential unmovable

The values used with each access method are listed below.

DSORG must always be coded in the DCB macro instruction, and, with certain access methods, must be coded on the DD statement.

BDAM -- DA or DAU (PS or PSU when creating the data set). The DSORG subparameter must be coded on the DD statement that defines the data set. When creating the data set, the DSORG subparameter must be coded as DA or DAU on the DD statement that defines the data set and PS or PSU in the DCB macro instruction.

BISAM -- IS; must be coded on the DD statement.

BPAM -- PO or POU

BSAM -- PS or PSU

BTAM -- CX

EXCP -- PS, PO, DA, or IS

GAM -- GS

QISAM -- IS or ISU (ISU can be specified only when creating the data set). The DSORG subparameter must be coded on the DD statement that defines the data set.

QSAM -- PS or PSU

QTAM -- MQ, CQ, or CX

EROPT= {ACC / SKP / ABE} Can be used with QSAM

Specifies the option to be executed if an error occurs in writing or reading a record, as follows:

ACC -- Accept the block causing the error.

SKP -- Skip the block causing the error (implies RELSE).

ABE -- Cause abnormal end of task.

If the subparameter is not specified by any source, ABE is assumed.

FUNC=function Can be used with BSAM, QSAM

Specifies the type of data set to be opened for the 3525 Card Read-Punch-Print, as follows:

I - interpret punch data set
R - read
P - punch
W - print
D - data protection for a punch data set
X - printer
T - two-line printer

The only valid combinations of these values are:

I	WT	RWX	PWXT
R	WXT	RWT	RPW
P	RP	RWXT	RPWX
W	RPD	PW	RPWXT
WX	RW	PWX	RPWD

If this information is not supplied by any source, the system assumes P.

Note that D, X, and T cannot be coded alone. If you specify D, the data protection image (DCI) must be stored in SYS1.IMAGELIB and you must code the image-identifier for the data protection image in the FCB parameter. For more detailed information, see the publication OS Data Management Services, GC26-3746.

GNCP=number — Can be used with GAM

Specifies the maximum number of input/output macro instructions that will be issued before a WAIT macro instruction. The value of GNCP must be from 1 to 99 at execution time. If the value of GNCP is not specified by any source, a value of 1 is assumed. The subparameters GNCP, BFTEK, BFALN, and HIARCHY are mutually exclusive subparameters. Therefore, if GNCP is coded, do not code any of these other subparameters. For additional information on the GNCP subparameter, refer to the publication IBM System/360 Operating System: Graphic Programming Services for IBM 2250 Display Unit, GC27-6909.

HIARCHY={0|1} — Can be used with BDAM, BISAM, BPAM, BSAM, EXCP, QISAM, QSAM

Specifies the storage hierarchy in which the buffer pool is to be formed as follows:

0 -- forms the pool from available space in processor storage.
1 -- forms the pool from available space in IBM 2361 Core Storage.

If the HIARCHY subparameter is not specified by any source, and if a hierarchy designation is not supplied by the GETPOOL macro instruction, hierarchy 0 is assumed.

The buffer pool is formed in the user partition or region within the indicated hierarchy. If space is unavailable within the hierarchy specified, the task is abnormally terminated.

INTVL=number — Can be used with QTAM

Specifies the polling interval (i.e., the number of seconds of intentional delay between passes through a polling list) for the lines in this line group. After all the terminals in a polling list for a given line have been polled (beginning to end), a delay equal to the number of seconds specified in this subparameter occurs before polling is restarted at the beginning of the list. The number specified must not be greater than 255.

If this subparameter is not specified by any source, INTVL=0 is assumed. This subparameter must be omitted if the line group consists of switched lines, WTTA lines, or if the Auto Poll feature is used.

KEYLEN=number — Can be used with BDAM, BPAM, BSAM, EXCP, QISAM (output only)

Specifies the length, in bytes, of the keys used in the data set. Except for QISAM, the keys are associated with blocks on direct access devices; the keys for indexed sequential data sets are associated with records. The maximum key length is always 255 bytes.

The subparameters KEYLEN, CODE, MODE, PRTSP, STACK, and TRTCH are mutually exclusive subparameters. Therefore, if KEYLEN is coded, do not code any of these other subparameters.

Note for BDAM: If standard labels are used, the key length information can be supplied from the data set label for an existing data set. If a key length is not supplied by any source, no input or output requests that require a key may be issued.

Note for BPAM and BSAM: If standard labels are used, the key length information can be supplied from the data set label for an existing data set. If a key length is not supplied by any source before the

Macro instruction is issued, a length of zero (no keys) is assumed.

Note for QISAM: For an existing data set with standard labels, the key length can only be supplied from the data set label.

LIMCT=number — Can be used with BDAM

Specifies the number of blocks, if relative block addressing is used, or the number of tracks, if relative track addressing is used, that are to be searched for a block or available space when the extended search option (OPTCD=E) is specified. The number may equal or exceed the number of blocks or tracks in the data set, in which case the entire data set is searched.

If the extended search option is not specified, the LIMCT subparameter is ignored.

LRECL=number — Can be used with BPAM, BSAM, QISAM (output only), QSAM, TCAM

Specifies the actual or maximum length, in bytes, of a logical record. The record length is required for fixed-length and variable-length records; for variable-length records, the maximum record length should be specified. The length cannot exceed the block size (BLKSIZE) value except for variable-length spanned records.

- If RECFM=V or VB, then LRECL must be equal to the maximum record length +4.
- If RECFM=F or FB, then LRECL must be equal to the logical record length.
- If RECFM=U, then LRECL should be omitted.
- If RECFM=D or DB, then LRECL must be equal to the maximum record length +4.

Note for BPAM: The record length is required for fixed-length records only.

Note for BSAM: The record length can be omitted from all sources, in which case the block size specification (BLKSIZE) is used. For variable-length spanned records (VS or VBS) processed under BSAM, if logical record exceeds 32,756, specify LRECL=X. For ASCII records on magnetic tape, the maximum record length is 2048 bytes and the minimum record length is 18 bytes.

Note for QISAM: For unblocked records, with a relative key position (RKP) of zero, the record length includes only the data portion of the record. The record length can be specified only when creating the data set.

Note for QSAM: For variable-length spanned records (VS or VBS) processed under QSAM (locate mode), if logical record exceeds 32,756, specify LRECL=X. For ASCII records on magnetic tape, the maximum record length is 2048 bytes and the minimum record length is 18 bytes.

Note for TCAM: The record length should include the source and control bytes if these are specified by the OPTCD suboperands. The record length is required for fixed-length records only.

MODE= $\left\{\begin{matrix} C & \left[\begin{matrix} O \\ R \end{matrix}\right] \\ E & \left[\begin{matrix} O \\ R \end{matrix}\right] \end{matrix}\right\}$ — Can be used with BSAM, EXCP, QSAM

Specifies the mode of operation to be used with a card reader, a card punch, a card read-punch, or a card read-punch-print, as follows:

C - card image (column binary) mode
E - EBCDIC mode
O - optical mark read mode
R - read column eliminate mode

Note: Do not code the MODE subparameter for data entered through the input stream.

If this information is not supplied by any source, the system assumes E.

If you specify O or R, the system assumes that the first card of the data set is the format descriptor. For more information on the format descriptor, refer to the publication OS Data Management Services, GC26-3746.

The DCB subparameters MODE, CODE, KEYLEN, PRTSP, and TRTCH are mutually exclusive. Therefore, if you code MODE, do not code any of these other subparameters.

NCP=number — Can be used with BISAM, BPAM, BSAM

Specifies the maximum number of READ or WRITE macro instructions issued before a CHECK macro instruction is issued. The maximum number allowed is based on limits established when the system was generated. If chained scheduling is used, NCP must be specified as more than 1.

If not specified by any source, 1 is assumed.

NTM=number — Can be used with QISAM

Specifies the number of tracks to be used for a cylinder index. When the specified number of tracks has been filled, a master index is created. This information is required only when the master index option (OPTCD=M) is selected.

If not specified by any source and OPTCD=M is specified, the master index option is ignored.

OPTCD= (A B C E F H I L M O Q R T U W Y Z) — Can be used with BDAM, BPAM, BSAM, EXCP, QISAM (output only), QSAM, TCAM

Specifies the optional services to be performed by the control program. All optional services must be requested by the same source. The characters may be coded in any order and, when used in combination, no commas are permitted between characters.

A -- Actual device addresses are to be presented ("block address" operand) in READ and WRITE macro instructions. For BDAM, R requests the same option as A, and either can be coded.

B -- Requests that end-of-file recognition be disregarded for tapes. This is to prevent premature end-of-file indication for multi-volume input data sets on tape when such tapes have end-of-file labels on volumes before the last volume.

C -- For BPAM,BSAM,QSAM: requests that chained scheduling be used.

For TCAM: specifies that one byte of the work area be used to indicate if a segment of a message is the first, intermediate, or last segment.

E -- An extended search (more than one track) is to be performed for a block or available space. (The LIMCT subparameter must also be specified; otherwise, this option is ignored.)

F -- Feedback may be requested in READ and WRITE macro instructions and the device address returned is to be of the form presented to the control program.

H -- For Optical Readers (BSAM): requests hopper empty exit.

For DOS Tape Input Files (BSAM and QSAM): requests the system to check for and bypass any DOS checkpoint records found on the tape.

I -- Requests that the control program use the independent overflow areas for overflow records.

L -- Requests that the control program delete records that have a first byte of all ones; records so marked may be deleted when space is required for new records. Do not specify this option for blocked records if RKP=0.

M -- Requests that master indexes be created as required, according to the information in the NTM subparameter. This option is ignored if the subparameter NTM=number is not specified.

O -- Requests online correction for Optical Readers (QSAM).

Q -- Specifies that translation from ASCII input to EBCDIC is required or that translation from EBCDIC to ASCII output is required. If the subparameter AL or AUL is coded in the LABEL parameter of the DD statement, the OPTCD subparameter will be treated as if Q was specified.

R -- For BDAM, actual device addresses are to be presented ("block address" operand) in READ and WRITE macro instructions.

For QISAM, requests the control program to place reorganization criteria information in the RORG1, RORG2, and RORG3 fields of the data control block. This option is provided whenever the OPTCD subparameter is omitted from all sources.

T -- Requests user totaling facility.

U -- For BSAM,QSAM: Only for 1403 printers with the Universal Character Set feature. Unblocks data checks and allows analysis by an appropriate error analysis (SYNAD) routine. If U is omitted, data checks are blocked (not recognized as errors).

When coding OPTCD=U, specify a printer in your program. Do not use the SYSOUT parameter unless the operator specifies a direct system output (DSO) writer.

For ISAM: specifies the full track index write feature.

For TCAM: specifies that the work unit to be handled is a message. If U is omitted, the work unit is assumed to be a record.

W -- For BDAM,BPAM,BSAM,QSAM: Requests a validity check for write operations on direct access devices. If the device is a 2321 data cell, validity checking is always performed whether requested or not.

For TCAM: Specifies that the name of each message source is to be placed in an eight-byte field in the work area.

Y -- Requests that the control program use the cylinder overflow areas for overflow records.

Z -- For input from a magnetic tape: Requests the control program to shorten its normal error recovery procedure. When Z is specified, a data check is considered permanent after five unsuccessful attempts to read a record. This option is available only if selected at system generation. It should be used only when a tape is known to be faulty and there is no need to process every record. The error analysis (SYNAD) routine should keep a count of the number of permanent errors, and should terminate processing if the number becomes excessive.

For input from a direct access storage device (DASD): Specifies search direct (SD) for sequential data sets.

Only certain options can be selected with each access method, as follows:

For BDAM:

A or R E F W

For BPAM:

C W WC

For BSAM and QSAM:

B C H Q T U W Z UC WC ZC

For EXCP:

Z

For QISAM:

I L M R U W Y

For BISAM:

L

For QSAM only:

O

For BSAM only:

H

For TCAM:

C U W

PCI=([N R A] [,N ,R ,A]) Can be used with TCAM

specifies if and how a program-controlled interruption (PCI) is to be used to control the allocating and freeing of buffers. The suboperands apply to receiving and sending operations respectively.

N -- specifies that no PCIs are taken during filling (on receiving operations) or emptying (on sending operations) of buffers. Buffers are freed at the end of transmission.

R -- specifies that after the first buffer is filled (on receiving operations) or emptied (on sending operations), a PCI occurs during the filling or emptying of each succeeding buffer. The completed buffer is freed, but no new buffer is allocated to take its place.

A -- specifies that after the first buffer is filled (on receiving operations) or emptied (on sending operations), a PCI occurs during the filling or emptying of the next buffer. The first buffer is freed. A buffer is allocated in place of the freed buffer.

If this operand is not specified by any source, PCI=(A,A) is assumed.

PRTSP= {0 1 2 3} Can be used with BSAM, EXCP, QSAM

Specifies the line spacing on a printer as 0, 1, 2, or 3 lines between printout. This subparameter is valid only if control characters are not present (A or M is not specified in the RECFM subparameter).

If not supplied by any source, 1 is assumed.

The subparameters PRTSP, CODE, KEYLEN, MODE, STACK, and TRTCH are mutually exclusive subparameters. Therefore, if PRTSP is coded, do not code any of these other subparameters.

RECFM=type Can be used with BDAM, BPAM, BSAM, EXCP, QISAM (output only), QSAM, TCAM

Specifies the format and characteristics of the records in the data set. The format and characteristics must be completely described by one source.

If this subparameter is omitted, an undefined-length record is assumed with no optional features provided, except for QISAM where variable-length records are assumed, and QTAM where a message segment is assumed.

Both the record format and characteristics are specified using the characters defined below. The allowable combinations of characters are indicated for the associated access methods; for increased efficiency, the characters should be coded in the order shown. However, alphabetical order is not required.

Character Definitions

A The record contains ASA printer control characters.

B The records are blocked.

D The ASCII records are of variable length. Each record on magnetic tape has a four-byte record descriptor field giving the record length in decimal.

F The records are of fixed length.

G The message data provided in the work unit is a complete message.

M The records contain machine code control characters.

R The message data provided in the work unit is a complete record.

S For fixed-length records, the records are written as standard blocks, i.e., no truncated blocks or unfilled tracks within the data set, with the exception of the last block or track.

For variable-length records, a record may span more than one block. Exchange buffering (BFTEK=E) cannot be specified.

For QTAM, the message data provided in the work unit is a message segment.

T The records may be written onto overflow tracks if required. Exchange buffering (BFTEK=E) or chained scheduling (OPTCD=C) should not be used because they will be ignored.

U The records are of undefined length.

V The records are of variable length. (Variable length records cannot be in ASCII.)

Only certain characters and combinations of characters can be selected with each access method. The allowable combinations of characters are indicated for the associated access methods; for increased efficiency, the characters should be coded in the order shown.

For BDAM:

```
{ U       }
{ V       }
{ F [T]   }
```

For BPAM:

```
{ U     [T]  [A]  }
{            [M]  }
{ V  [B]     [A]  }
{    [T]     [M]  }
{    [BT]         }
{ F  [B]     [A]  }
{    [T]     [M]  }
{    [BT]         }
```

For BSAM and QSAM:

```
{ U     [T]  [A]  }
{            [M]  }
{ V  [B]          }
{    [S]          }
{    [T]          }
{    [BS]         }
{    [BT]         }
{    [ST]    [A]  }
{    [BST]   [M]  }
{ F  [B]          }
{    [S]          }
{    [T]          }
{    [BS]         }
{    [BT]         }
{    [ST]    [A]  }
{    [BST]   [M]  }
```

For BSAM and QSAM using ASCII data sets on tape:

```
{ D     [B]  [A]  }
{ U          [A]  }
{ F     [B]  [A]  }
```

Note: A or M cannot be specified if the PRTSP subparameter is specified.

For QISAM:

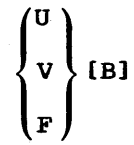
```
{ V  [B] }
{ F  [B] }
```

For QTAM:

```
{ G }
{ R }
{ S }
```

DD

For TCAM:



RESERVE=(number1,number2) Can be used with TCAM

specifies the number of bytes (from 0 to 255) to be reserved in a buffer for insertion of data by the DATETIME and SEQUENCE macros. number1 indicates that space is to be reserved in the first buffer of each incoming message; number2, that space is to be reserved in all buffers except the first. If RESERVE is not coded, no space is reserved.

RKP=number Can be used with QISAM (output only)

Specifies the position of the first byte of the record key, relative to the beginning of each record. (The beginning byte of a record is addressed as 0.)

If RKP=0 is specified for blocked fixed-length records, the key begins in the first byte of each record, and the delete option (OPTCD=L) must not be specified. If RKP=0 is specified for unblocked fixed-length records, the key is not written in the data field; the delete option can be specified.

For variable-length records, the relative key position must be 4 or greater, when the delete option (OPTCD=L) is not specified. The relative key position must be 5 or greater if the delete option is specified.

If this information is not specified by any source, a relative key position of zero is assumed.

SOWA=number Can be used with QTAM

Specifies the size, in bytes, of the user-provided input work areas. The value must be less than 32,768 and must include the 4-byte user prefix.

If this subparameter is not specified by any source, the telecommunications job, when executed, is terminated.

STACK= {1 | 2} Can be used with BSAM, EXCP, QSAM

Specifies the stacker bin to receive the card, and is either 1 or 2.

If not specified by any source, 1 is assumed.

The subparameters STACK, CODE, KEYLEN, PRTSP, and TRTCH are mutually exclusive subparameters. Therefore, if STACK is coded, do not code any of these other subparameters.

THRESH=number Can be used with TCAM

specifies the percentage of the nonreusable disk message queue records to be used before a flush closedown occurs. If this operand is omitted, closedown occurs when 95% of the records have been used.

TRTCH= {C / E / T / ET} Can be used with BSAM, EXCP, QSAM

Specifies the recording technique for seven-track tape. (This parameter is only to be used with seven track tape.)

C Data conversion feature is to be used, with odd parity and no translation.

E Even parity, with no translation and no conversion.

T Odd parity and no conversion, and BCD to EBCDIC translation is required when reading; EBCDIC to BCD translation when writing.

ET Even parity and no conversion, and BCD to EBCDIC translation is required when reading; EBCDIC to BCD translation when writing.

If this subparameter is not specified by any source, odd parity and no translation or data conversion is assumed.

The subparameters TRTCH, CODE, KEYLEN, MODE, PRTSP, and STACK are mutually exclusive subparameters. Therefore, if TRTCH is coded, do not code any of these other subparameters.

Examples of the DCB Parameter

1.
```
//DD1      DD   DSNAME=ALP,DISP=(,KEEP),VOLUME=SER=44321,          X
//             UNIT=2400,DCB=(RECFM=FB,LRECL=240,BLKSIZE=960,      X
//             DEN=1,TRTCH=C)
```

This DD statement defines a new data set and contains the information necessary to complete the data control block.

2.
```
//DD2      DD   DSNAME=BAL,DISP=OLD,DCB=(RECFM=F,LRECL=80,         X
//             BLKSIZE=80)
//DD3      DD   DSNAME=CNANN,DISP=(,CATLG,DELETE),UNIT=2400,       X
//             LABEL=(,NL),VOLUME=SER=663488,DCB=*.DD2
```

The statement named DD3 defines a new data set and requests the system to copy the DCB subparameters from the DD statement named DD2, which is in the same job step.

3.
```
//DD4      DD   DSNAME=JST,DISP=(NEW,KEEP),UNIT=2311,              X
//             SPACE=(CYL,(12,2)),DCB=(A.B.C,KEYLEN=8)
```

This DD statement defines a new data set and requests the system to copy DCB information from the data set label of the cataloged data set named A.B.C. If the data set label contains a key length specification, it is overridden since KEYLEN is coded on the DD statement.

```
4.  //DD5      DD   DSNAME=SAME,DISP=OLD,UNIT=2311,                    X
    //               DCB=(*.STEP1.PROCSTP5.DD8,BUFNO=5)
```

This DD statement defines an existing data set and requests the system to copy the DCB subparameters from the DD statement named DD8, which is contained in the procedure step named PROCSTP5. The cataloged procedure was called by the job step named STEP1. If any of the DCB subparameters coded on the procedure DD statement have been previously defined for this data set, they are ignored. If the BUFNO subparameter has not been previously specified for the data set, then five buffers are assigned to the data control block.

DCB Subparameter	ALGOL	Assembler	COBOL E	COBOL F	ANS COBOL [3]	FORTRAN E	FORTRAN G & H	PL/I F
BFALN=		F or D [1,2]						
BFTEK=		S or E (QSAM only) [1,2]						
BLKSIZE=	number of bytes	number of bytes [1]		number of bytes [1]	number of bytes [1]	number of bytes	number of bytes	number of bytes [1]
BUFL=		number of bytes [1]						
BUFNO=		number of buffers [1]	number of buffers [1]	number of buffers [1]	number of buffers [1]	1 or 2	1 or 2	number of buffers [1]
DIAGNS=	TRACE	TRACE	TRACE	TRACE	TRACE	TRACE	TRACE	TRACE
EROPT=		ABE [1]	ABE	ABE	ABE			
FUNC=		any of possible combinations [4]	any of possible combinations [4]	any of possible combinations [4]	any of possible combinations [4]	any of possible combinations [4]	any of possible combinations [4]	any of possible combinations [4]
HIARCHY=		0 or 1 [1]						
LRECL=	number of bytes	number of bytes [1]				number of bytes	number of bytes	number of bytes [1]
MODE=		C[R] or E[R] [1]	C[R] or E[R]	C[R] or E[R]	C[R] or E[R]	C[R] or E[R]	C[R] or E[R]	C[R] or E[R]
NCP=		number of channel programs (BSAM only) [1]						
OPTCD=		[C] [1]	[C]	[C]	[C]	[C]	[C]	[C]
RECFM=	E[B][A]	U[A/M] or V[B/S/BS][A/M] or F[B/S/BS][A/M] [1]				U[A] or F[B][A/M]	U[A/M] or F[B][A/M]	U[A/M] or V[B/S/BS][A/M] or F[B/S/BS][A/M]
STACK=		1 or 2 [1]	1 or 2	1 or 2	1 or 2	1 or 2	1 or 2	1 or 2

[1] This function can be specified in your program rather than in the DD statement.

[2] For QSAM, you must specify both BFALN and BFTEK on the DD statement, or omit both.

[3] American National Standard COBOL.

[4] See the description of the FUNC subparameter in the Glossary of DCB Subparameters for a list of possible combinations.

Figure 10. DCB Subparameters for Card Punch

DCB Subparameter	ALGOL	Assembler	COBOL E	COBOL F	ANS COBOL [4]	FORTRAN E	FORTRAN G & H	PL/I F
BFALN=		F or D [1,2]						
BFTEK=		S or E (QSAM only) [1,2]						
BLKSIZE=	number of bytes	number of bytes [1]		number of bytes [1]	number of bytes [1]	number of bytes	number of bytes	number of bytes [1]
BUFL=		number of bytes [1]						
BUFNO=		number of buffers [1]	number of buffers [1]	number of buffers [1]	number of buffers [1]	1 or 2	1 or 2	number of buffers [1]
DIAGNS=	TRACE	TRACE	TRACE	TRACE	TRACE	TRACE	TRACE	TRACE
EROPT=		ACC or ABE [1]	ACC or ABE	ACC or ABE	ACC or ABE			
FUNC=		any of possible combinations [4]	any of possible combinations [4]	any of possible combinations [4]	any of possible combinations [4]	any of possible combinations [4]	any of possible combinations [4]	any of possible combinations [4]
HIARCHY=		0 or 1 [1]						
LRECL=	number of bytes	number of bytes [1]				number of bytes	number of bytes	number of bytes [1]
MODE=	C[R]or E[R]	C[R]or E[R]	C[R]or E[R]	C[R]or E[R]	C[R]or E[R]	C[R]or E[R]	C[R]or E[R]	C[R]or E[R]
NCP=		number of channel programs (BSAM only) [1]						number of channel programs (BSAM only) [1]
OPTCD=		[C] [U] [1]	[C]	[C]	[C]	[C]	[C]	[C] [U]
PRTSP=		0, 1, 2, or 3 [1,3]	0, 1, 2, or 3 [3]	0, 1, 2, or 3 [3]	0, 1, 2, or 3 [3]	0, 1, 2, or 3 [3]	0, 1, 2, or 3 [3]	0, 1, 2, or 3 [3]
RECFM=	F [B BS] [A]	U [A M] or V [B S BS] [A M] or F [B S BS] [A M] [1]				Formatted: U [A] or F [B] [A M] Unformatted: VS [B] [A M]	Formatted: U [A M] or F [B] [A M] Unformatted: VS [B] [A M]	U [A M] or V [B S BS] [A M] or F [B S BS] [A M] [1]

[1] This function can be specified in your program rather than in the DD statement.

[2] For QSAM, you must specify both BFALN and BFTEK on the DD statement, or omit both.

[3] Do not use if A or M is specified in the RECFM.

[4] American National Standard COBOL.

[5] See the description of the FUNC subparameter in the Glossary of DCB Subparameters for a list of possible combinations.

Figure 11. DCB Subparameters for Printer

DCB Subparameter	ALGOL	Assembler	COBOL E	COBOL F	ANS COBOL [4]	FORTRAN E	FORTRAN G & H	PL/I F
BFALN=		F or D [1,2]						
BFTEK=		S, E, or A (QSAM only) [1,2]						
BLKSIZE=	number of bytes	number of bytes [1]		number of bytes [1]	number of bytes [1]	number of bytes	number of bytes	number of bytes [1]
BUFL=		number of bytes [1]						
BUFNO=		number of buffers [1]	number of buffers [1]	number of buffers [1]	number of buffers [1]	1 or 2	1 or 2	number of buffers [1]
BUFOFF= [5]		n or L	n or L	n or L	n or L	n or L	n or L	n or L
DEN=		0,1,2, or 3 [1,3]	0,1, or 2 [3]	0,1,2, or 3 [3]	0,1,2, or 3 [3]	0,1,2, or 3 [3]	0,1,2, or 3 [3]	0,1,2, or 3 [3]
DIAGNS=	TRACE	TRACE	TRACE	TRACE	TRACE	TRACE	TRACE	TRACE
EROPT=		ABE [1]	ABE	ABE	ABE			
HIARCHY=		0 or 1 [1]						
LRECL=	number of bytes	number of bytes [1]				number of bytes	number of bytes	number of bytes [1]
NCP=		number of channel programs (BSAM only) [1]						number of channel programs (BSAM only) [1]
OPTCD=		[C][T] [Q] (ASCII) [1]	[C] [Q] (ASCII)	[C] [Q] (ASCII)	[C]	[C]	[C] [Q] (ASCII)	[C] [Q] (ASCII) [1]
RECFM=	F[B/BS] [A] [D]	U[A/M] or V[B/S/BS][A/M] or F[B/S/BS][A/M] or D[B] [A] [1]		D		Formatted: U[A] V[B] [A] or F[B][A/M] D[B] [A] Unformatted: VS[B][A/M]	Formatted: U[A/M] or V[B][A/M] or F[B][A/M] D[B][A] Unformatted: VS[B][A/M]	U[A/M] or V[B/S/BS][A/M] or F[B/S/BS][A/M] or D[B] [1]
TRTCH=		C, E, ET, or T [1]	C, E, ET, or T	C, E, ET, or T	C, E, ET, or T	C, E, ET, or T	C, E, ET, or T	C, E, ET, or T

[1] This function can be specified in your program rather than in the DD statement.

[2] For QSAM, you must specify both BFALN and BFTEK on the DD statement or omit both.

[3] If DEN is omitted, see chart on page 137 for assumed values.

[4] American National Standard COBOL.

[5] This parameter is specified only for ASCII data sets on magnetic tape.

Figure 12. DCB Subparameters for Creating a Data Set on Magnetic Tape

DCB Subparameter	ALGOL	Assembler	COBOL E	COBOL F	ANS COBOL [3]	FORTRAN E	FORTRAN G & H	PL/I F
BFALN=		F or D [1,2]						
BFTEK=		S, E, or A (QSAM only) [1,2]						
BLKSIZE=	number of bytes	number of bytes [1]		number of bytes [1]	number of bytes [1]	number of bytes	number of bytes	number of bytes [1]
BUFL=		number of bytes [1]						
BUFNO=		number of buffers [1]	number of buffers [1]	number of buffers [1]	number of buffers [1]	1 or 2	1 or 2	number of buffers [1]
DIAGNS=	TRACE	TRACE	TRACE	TRACE	TRACE	TRACE	TRACE	TRACE
DSORG=		PS or PSU [1]						
EROPT=		ABE [1]	ABE	ABE	ABE			
HIARCHY=		0 or 1 [1]						
KEYLEN=		number of bytes [1]						number of bytes [1]
LRECL=	number of bytes	number of bytes [1]				number of bytes	number of bytes	number of bytes [1]
NCP=		number of channel programs (BSAM only) [1]						number of channel programs (BSAM only) [1]
OPTCD=		[W] [C] [T] [1]	[W] [C]	[W] [C]	[W] [C]	[C]	[C]	[W] [C]
RECFM=	F[B/BS][A][T]	U[T][A/M] or V[B/S/T/BS/BT/ST/BST][A] or [M] or F[B/S/T/BS/BT/ST/BST][A/M]				Formatted: U[A][T] or V[B][A][T] or F[B][A/M][T] Unformatted: VS[B][A/M][T]	Formatted: U[A/M][T] or V[B][A/M][T] or F[B][A/M][T] Unformatted: VS[B][A/M][T]	[1] U[A/M] or V[B/S/BS][A/M] or F[B/S/BS][A/M]

[1] This function can be specified in your program rather than in the DD statement.

[2] For QSAM, you must specify both BFALN and BFTEK on the DD statement or omit both.

[3] American National Standard COBOL.

Figure 13. DCB Subparameters for Creating a Sequential Data Set on Direct Access Devices

DCB Subparameter	Assembler	COBOL E	COBOL F	ANS COBOL [2]	FORTRAN E	FORTRAN G & H	PL/I F
BFALN=	F or D [1]						
BFTEK=	R [1]						
BLKSIZE=	number of bytes [1]						number of bytes [1]
BUFL=	number of bytes [1]						
BUFNO=	number of buffers [1]				1 or 2	1 or 2	
DIAGNS=	TRACE	TRACE	TRACE	TRACE	TRACE	TRACE	TRACE
DSORG=	DA or DAU [1]		DA	DA	DA	DA	DA
HIARCHY=	0 or 1 [1]						
KEYLEN=	number of bytes [1]						number of bytes
LIMCT=	number of tracks or blocks [1]			number of tracks or blocks [1]			number of tracks or blocks
OPTCD=	[W] [E] [F] [R A] [1]	[W]	[W]	[W] [E]			[W]
RECFM=	U,V, or F[T] [1]						U,V, or F[T] [1]

[1] This function can be specified in your program rather than in the DD statement.

[2] American National Standard COBOL.

Figure 14. DCB Subparameters for Creating a Direct Data Set

DCB Subparameter	Assembler [1]
BFALN=	F or D [1]
BLKSIZE=	number of bytes [1]
BUFL=	number of bytes [1]
BUFNO=	number of buffers [1]
DIAGNS=	TRACE
HIARCHY=	0 or 1 [1]
KEYLEN=	number of bytes [1]
LRECL=	number of bytes [1]
NCP=	number of channel programs [1]
OPTCD=	[W] [C] [1]
RECFM=	U[T][A/M] or V[B/T/BT][A/M] or F[B/T/BT][A/M] [1]

[1] This function can be specified in your program rather than in the DD statement.

Figure 15. DCB Subparameters for Creating a Partitioned Data Set

DCB Subparameter	ALGOL	Assembler	COBOL E	COBOL F	ANS COBOL [3]	FORTRAN E	FORTRAN G & H	PL/I F
BFALN=		F or D [1,2]						
BFTEK=		S or E (QSAM only) [1,2]						
BLKSIZE=	number of bytes	number of bytes [1]		number of bytes [1]	number of bytes [1]	number of bytes	number of bytes	number of bytes [1]
BUFL=		number of bytes [1]						
BUFNO=		number of buffers [1]	number of buffers [1]	number of buffers	number of buffers	1 or 2	1 or 2	number of buffers [1]
DIAGNS=	TRACE	TRACE	TRACE	TRACE	TRACE	TRACE	TRACE	TRACE
EROPT=		ABE [1]	ABE	ABE	ABE			
FUNC=		any of possible combinations [4]	any of possible combinations [4]	any of possible combinations [4]	any of possible combinations [4]	any of possible combinations [4]	any of possible combinations [4]	any of possible combinations [4]
HIARCHY=		0 or 1 [1]						
LRECL=	number of bytes	number of bytes [1]				number of bytes	number of bytes	number of bytes [1]
MODE=		C[O/R] or E[O/R] [1]	C[O/R] or E[O/R]	C[O/R] or E[O/R]	C[O/R] or E[O/R]	C[O/R] or E[O/R]	C[O/R] or E[O/R]	C[O/R] or E[O/R] [1]
NCP=		number of channel programs (BSAM only) [1]						number of channel programs (BSAM only) [1]
OPTCD=		[C] [1]	[C]	[C]	[C]	[C]	[C]	[C]
RECFM=	E[B][A]	U[A/M] or V[B/S/BS][A/M] or F[B/S/BS][A/M] [1]				U[A] or F[B][A/M]	U[A/M] or F[B][A/M]	U[A/M] or V[B/S/BS][A/M] or F[B/S/BS][A/M] [1]
STACK=		1 or 2 [1]	1 or 2	1 or 2	1 or 2	1 or 2	1 or 2	1 or 2

[1] This function can be specified in your program rather than in the DD statement.

[2] For QSAM, you must specify both BFALN and BFTEK on the DD statement, or omit both.

[3] American National Standard COBOL.

[4] See the description of the FUNC subparameter in the Glossary of DCB Subparameters for a list of possible combinations.

Figure 16. DCB Subparameters for Card Reader

DCB Subparameter	ALGOL	Assembler	COBOL E	COBOL F	ANS COBOL [3]	FORTRAN E	FORTRAN G & H	PL/I F
BFALN=		F or D [1,2]						
BFTEK=		S or E (QSAM only) [1,2]						
BLKSIZE=	number of bytes	number of bytes [1]		number of bytes [1]	number of bytes [1]	number of bytes	number of bytes	number of bytes [1]
BUFL=		number of bytes [1]						
BUFNO=		number of buffers [1]	number of buffers [1]	number of buffers	number of buffers	1 or 2	1 or 2	number of buffers [1]
CODE=		I,F,B,C,A,T, or N [1]						I,F,B,C,A,T or N [1]
DIAGNS=	TRACE	TRACE	TRACE	TRACE	TRACE	TRACE	TRACE	TRACE
EROPT=		ABE [1]	ABE	ABE	ABE			
HIARCHY=		0 or 1 [1]						
LRECL=	number of bytes	number of bytes [1]				number of bytes	number of bytes	number of bytes [1]
NCP=		number of channel programs (BSAM only) [1]						
OPTCD=		[C] [1]	[C]	[C]	[C]	[C]	[C]	[C]
RECFM=	E[B][A]	U[A/M] or V[B/S/BS][A/M] or F[B/S/BS][A/M] [1]				U[A] or V[B][A] or F[B][A/M]	U[A/M] or V[B][A/M] or F[B][A/M]	U or F [1]

[1] This function can be specified in your program rather than in the DD statement.

[2] For QSAM, you must specify both BFALN and BFTEK on the DD statement, or omit both.

[3] American National Standard COBOL.

Figure 17. DCB Subparameters for Paper Tape Reader

DCB Subparameter	ALGOL	Assembler	COBOL E	COBOL F	ANS COBOL [3]	FORTRAN E	FORTRAN G & H	PL/I F
BFALN=		F or D [1,2]						
BFTEK=		S, E, or A (QSAM only) [1,2]						
BUFL=		number of bytes [1]						
BUFNO=		number of buffers [1]	number of buffers [1]	number of buffers [1]	number of buffers [1]	1 or 2	1 or 2	number of buffers [1]
BUFOFF [4]		n or L	n or L	n or L	n or L	n or L	n or L	n or L
DIAGNS=	TRACE	TRACE	TRACE	TRACE	TRACE	TRACE	TRACE	TRACE
EROPT=		ABE,ACC, or SKP [1]	ABE,ACC, or SKP	ABE,ACC, or SKP	ABE,ACC, or SKP			
HIARCHY=		0 or 1 [1]						
NCP=		number of channel programs (BSAM only) [1]						number of channel programs (BSAM only) [1]
OPTCD=		[C] [Z] [B] [Q] (ASCII) [1]	[C] [Q] (ASCII)	[C] [Q] (ASCII)	[C]	[C]	[C] [Q] (ASCII)	[C] or [Z] [Q] (ASCII)

[1] This function can be specified in your program rather than in the DD statement.

[2] For QSAM, you must specify both BFALN, and BFTEK on the DD statement, or omit both.

[3] American National Standard COBOL.

[4] This parameter is specified only for ASCII data sets on magnetic tape.

Figure 18. DCB Subarameters for Retrieving a Data Set on Magnetic Tape

DCB Subparameter	ALGOL	Assembler	COBOL E	COBOL F	ANS COBOL [3]	FORTRAN E	FORTRAN G & H	PL/I F
BFALN=		F or D [1,2]						
BFTEK=		S, E, or A (QSAM only) [1,2]						
BUFL=		number of bytes [1]						
BUFNO=		number of buffers [1]	number of buffers [1]	number of buffers [1]	number of buffers [1]	1 or 2	1 or 2	number of buffers [1]
DIAGNS=	TRACE	TRACE	TRACE	TRACE	TRACE	TRACE	TRACE	TRACE
EROPT=		ABE,ACC, or SKP [1]	ABE,ACC, or SKP	ABE,ACC, or SKP				
HIARCHY=		0 or 1 [1]						
NCP=		number of channel programs (BSAM only) [1]						

[1] This parameter can be specified in your program rather than in the DD statement.

[2] For QSAM, you must specify both BFALN and BFTEK on the DD statement, or omit both.

[3] American National Standard COBOL.

Figure 19. DCB Subparameters for Retrieving a Sequential Data Set on Direct Access Device

DCB Subparameter	Assembler	COBOL E	COBOL F	ANS COBOL [2]	FORTRAN E	FORTRAN G	PL/I F
BFALN=	F or D [1]						
BFTEK=	R [1]						
BUFL=	number of bytes [1]						
BUFNO=	number of buffers [1]				1 or 2	1 or 2	number of buffers
DIAGNS=	TRACE	TRACE	TRACE	TRACE	TRACE	TRACE	TRACE
HIARCHY=	0 or 1 [1]						
LIMCT=	number of tracks or blocks [1]						number of tracks or blocks
OPTCD=	[E][F][R/A] [1]						

[1] This function can be specified in your program rather than in the DD statement.

[2] American National Standard COBOL.

Figure 20. DCB Subparameters for Retrieving a Direct Data Set

DCB Subparameter	Assembler
BFALN=	F or D [1]
BUFL=	number of bytes [1]
BUFNO=	number of buffers [1]
DIAGNS=	TRACE
HIARCHY=	0 or 1 [1]
NCP=	number of channel programs [1]
OPTCD=	[C] [1]

[1] This parameter can be specified in your program rather than in the DD statement.

Figure 21. DCB Subparameters for Retrieving a Partitioned Data Set

The DDNAME Parameter

```
DDNAME=ddname
```

ddname
the name of a following DD statement in the same job step that defines this data set.

Rules for Coding

1. The only parameters that can be coded with the DDNAME parameter are the DCB subparameters BLKSIZE, BUFNO and DIAGNS.

2. The DDNAME parameter cannot appear on a DD statement named JOBLIB.

3. You can code the DDNAME parameter up to five times in a job step or procedure step. However, each time the DDNAME parameter is coded, it must refer to a different ddname.

4. If the data set, which will be defined later in the job step, is to be concatenated with other data sets, the DD statements that define these other data sets must immediately follow the DD statement that includes the DDNAME parameter.

5. The DDNAME parameter must not be used to refer to a DD statement that has DYNAM coded on it.

6. A DD statement to which a DDNAME parameter refers cannot contain any reference to a DD statement that follows the one with the DDNAME parameter.

7. Do not code DSNAME and DDNAME parameters on the same DD statement.

What the DDNAME Parameter Does

The DDNAME parameter allows you to postpone defining a data set until later in the same job step. In the case of cataloged or in-stream procedures, this parameter allows you to postpone defining a data set in the procedure until the procedure is called by a job step.

The DDNAME parameter is most often used in cataloged procedures and in job steps that call procedures. It is used in cataloged or in-stream procedures to postpone defining data in the input stream until a job step calls the procedure. (Procedures cannot contain DD statements that define data in the input stream, i.e., DD * or DD DATA statements). It is used in job steps that call procedures to postpone defining data in the input stream on an overriding DD statement until the last overriding DD statement for a procedure step. (Overriding DD statements must appear in the same order as the corresponding DD statements in the procedure.)

When You Code the DDNAME Parameter

When the system encounters a DD statement that contains the DDNAME parameter, it saves the ddname of that statement. The system also temporarily saves the name specified in the DDNAME parameter so that it can relate that name to the ddname of a later DD statement. Once a DD statement with that corresponding name is encountered, the name is no longer saved. For example, if the system encounters this statement

```
//XYZ  DD  DDNAME=PHOB
```

the system saves XYZ and, temporarily, PHOB. Until the ddname PHOB is encountered in the input stream, the data set is a dummy data set.

When the system encounters a statement whose ddname has been temporarily saved, it does two things. It uses the information contained on this statement to define the data set; it associates this information with the name of the statement that contained the DDNAME parameter. The value that appeared in the DDNAME parameter is no longer saved by the system. To continue the above example, if the system encounters this statement

```
//PHOB   DD   DSNAME=NIN,DISP=(NEW,KEEP),UNIT=2400
```

the system uses the data set name and the disposition and unit information to define the data set; it also associates the ddname of the statement that contained the DDNAME parameter with this information. In this example, the ddname used is XYZ; the ddname PHOB is no longer saved. The data set is now defined, just as it would be if you had coded


```
//XYZ   DD   DSNAME=NIN,DISP=(NEW,KEEP),UNIT=2400
```

The system associates the ddname of the statement that contains the DDNAME parameter with the data set definition information. It does not use the ddname of the later statement that defines the data set. Therefore, any references to the data set, before or after the data set is defined, must refer to the DD statement that contains the DDNAME parameter, not the DD statement that defines the data set. The following sequence of control statements illustrates this:

```
//DD1      DD   DDNAME=LATER
              .
              .
              .
//LATER    DD   DSN=SET12,DISP=(NEW,KEEP),UNIT=2311,                X
//             VOLUME=SER=46231,SPACE=(TRK,(20,5))
              .
              .
              .
//DD12     DD   DSN=SET13,DISP=(NEW,KEEP),VOLUME=REF=*.DD1,         X
//             SPACE=(TRK,(40,5))
```

When you want to concatenate data sets, the unnamed DD statements must follow the DD statement that contains the DDNAME parameter, not the DD statement that defines the data set. The following sequence of control statements illustrates this:

```
//DDA      DD   DDNAME=DEFINE
//        DD   DSN=A.B.C,DISP=OLD
//        DD   DSN=SEVC,DISP=OLD,UNIT=2311,VOL=SER=52226
              .
              .
              .
//DEFINE   DD   *
         data
/*
```

You can use the DDNAME parameter up to five times in a job step or procedure step. However, each time the DDNAME parameter is coded, it must refer to a different ddname.

THE DCB SUBPARAMETERS BLKSIZE, BUFNO, AND DIAGNS

The DCB subparameters BLKSIZE, BUFNO, and DIAGNS can be coded with the DDNAME parameter. This allows you to assign these DCB characteristics to the data set defined in the referenced DD statement. When the DCB subparameters BLKSIZE, BUFNO, and DIAGNS are coded both on the DD statement that contains the DDNAME parameter and on the referenced DD statement, the subparameters coded on the former are ignored.

These subparameters would most often be coded with the DDNAME parameter when the referenced DD statement defines data in the input stream. Data in the input stream is written onto a direct access device, and the records are blocked as they are written. The input reader procedure normally assigns a block size and number of buffers for blocking. Coding the BLKSIZE subparameter allows you to specify that you want shorter blocks. Coding the BUFNO subparameter allows you to specify that you want fewer buffers. You cannot specify that you want larger blocks or more buffers than would be assigned by the input reader procedure. (When a job is submitted via remote job entry and the BUFNO subparameter is coded, the BUFNO subparameter is ignored.)

Examples of the DDNAME Parameter

1.
```
//STEP1    EXEC PGM=PROGRAM8
//DD1      DD   DDNAME=INPUT
//DD2      DD   DSNAME=WELL,DISP=OLD
```

The above statements make up the statements for a procedure step named STEP1, which is the first step of a procedure named MENT. The following statements illustrate how you would define DD1 as a data set in the input stream:

```
//STPA     EXEC PROC=MENT
//STEP1.INPUT  DD *
               .
               .
               data
               .
               .
/*
```

2.
```
//ST4      EXEC PGM=FIFTY
//DD1      DD   DDNAME=DD5
//DD2      DD   UNIT=2400
//DD3      DD   UNIT=2400
//DD4      DD   SYSOUT=B
//DD5      DD   DSNAME=ADDN,DISP=(,PASS),UNIT=2400
//ST5      EXEC PGM=FINE
//DD6      DD   DSNAME=*.ST4.DD1,DISP=(OLD,KEEP)
```

The DD statement named DD5 defines the data set for the statement named DD1. The DD statement of the second job step wants the system to obtain the data set name, unit and volume information of this data set. This is done by referring to the DD statement that contains the DDNAME parameter.

3.
```
//STEP8    EXEC PGM=BLOCK
//DD1      DD   DDNAME=SKIP
//        DD   DSNAME=A.B.C,DISP=OLD
//        DD   DSNAME=LEV.FIVE,DISP=OLD
//SKIP     DD   DSNAME=SEF,DISP=OLD,UNIT=2311,VOLUME=SER=111111
```

The DD statement named SKIP defines the data set for the statement named DD1. The two data sets, A.B.C. and LEV.FIVE, are concatenated with the data set named SEF.

4.
```
//STEPX    EXEC PGM=PROG12
//DD1      DD   DDNAME=LATER,DCB=(BLKSIZE=1600,BUFNO=2)
//DD2      DD   UNIT=2400
//DD3      DD   SYSOUT=F
//LATER    DD   *
           .
           .
           data
           .
           .
/*
```


The DD statement named LATER defines the data set for the statement named DD1. The DCB subparameters coded with the DDNAME parameter are used to block the input data.

5.
```
//STEPX    EXEC PGM=B403
//DDA      DD   DSNAME=SEL,DISP=OLD,VOLUME=SER=X3220,UNIT=2400
//DDB      DD   SYSOUT=B
```

The above statements make up the statements for a procedure step named STEPX, which is the first step of a procedure named TYPE. The following statements illustrate how you would use the DDNAME parameter when overriding both of the DD statements and the first overriding DD statement is to define data in the input stream:

```
//CALL          EXEC PROC=TYPE
//STEPX.DDA     DD   DDNAME=IN
//STEPX.DDB     DD   SYSOUT=G
//STEPX.IN      DD   *
                .
                .
                data
                .
                .
/*
```

6.
```
//MAR           EXEC PGM=DEPT12
//CARD1         DD   DDNAME=CARD4
//CARD2         DD   UNIT=2400
//CARD3         DD   DSNAME=NINE.SCR,DISP=OLD
//COGH          EXEC PGM=DEPT13
```

The DD statement named CARD1 contains the DDNAME parameter. This statement defines a dummy data set since there is no DD statement named CARD4 in the step.

The DISP Parameter

```
DISP=( [NEW ] [,DELETE ] [,DELETE ] )
       [OLD ] [,KEEP   ] [,KEEP   ]
       [SHR ] [,PASS   ] [,CATLG  ]
       [MOD ] [,CATLG  ] [,UNCATLG]
              [,UNCATLG]
              [,       ]
```

NEW
specifies that the data set is to be created in this job step.

OLD
specifies that the data set existed before this job step.

SHR
specifies that the data set existed before this job step and can be used simultaneously (shared) by another job, since it will only be read.

MOD
specifies that the read/write mechanism is to be positioned after the last record in the data set, and, if the system cannot find volume information for the data set, specifies that the data set is to be created.

,DELETE
specifies that the data set is no longer needed and its space on the volume is to be released at the end of this job step for use by other data sets.

,KEEP
specifies that the data set is to be kept on the volume at the end of this job step.

,PASS
specifies that the data set is to be passed for use by a subsequent job step in the same job.

,CATLG
specifies that the data set is to be kept at the end of this job step and an entry pointing to the data set is to be placed in the system catalog.

,UNCATLG
specifies that the data set is to be kept at the end of this job step but the entry pointing to the data set in the system catalog is to be removed.

,
specifies that a disposition is not explicitly specified for the data set, but a conditional disposition follows. A new data set is to be deleted and a data set that existed before execution of the job is to be kept at the end of this job step.

,DELETE
specifies that the data set is no longer needed and its space on the volume is to be released for use by other data sets if this step abnormally terminates.

,KEEP
specifies that the data set is to be kept on the volume if this step abnormally terminates.

,CATLG
specifies that an entry pointing to the data set is to be placed in the system catalog if this step abnormally terminates.

,UNCATLG
specifies that the entry pointing to the data set in the system catalog is to be removed if this step abnormally terminates.

Rules for Coding

1. If only the first subparameter is coded, you need not enclose it in parentheses.

2. If the data set is new, you can omit the subparameter NEW. However, if you specify a disposition or conditional disposition, you must code a comma to indicate the absence of NEW.

3. You can omit the DISP parameter if a data set is created and deleted during a job step.

4. If you do not want to change the automatic disposition processing performed by the system, you need not code the second subparameter. (When the second subparameter is not coded, the system automatically keeps data sets that did exist before the job and automatically deletes data sets that did not exist before the job.) If you omit the second subparameter and code a conditional disposition, you must code a comma to indicate the absence of the second subparameter.

5. The DISP, SYSOUT, and DDNAME parameters are mutually exclusive parameters; therefore, when SYSOUT or DDNAME is coded, do not code the DISP parameter.

6. You must specify a disposition of PASS or DELETE for a data set with a system-generated name: i.e., when DSNAME=dsname is omitted from the DD statement. Any other disposition will be overridden by the system with PASS.

WHAT THE DISP PARAMETER DOES

The DISP parameter describes to the system the status of a data set and indicates what is to be done with the data set after termination of the job step that processes it or at the end of the job. You can indicate in the DISP parameter one disposition to apply if the step terminates normally after execution and another to apply if the step terminates abnormally (conditional disposition).

Specifying the Data Set's Status

A data set is either a new data set or an existing data set. What you plan to do with the data set determines which status you code as the first subparameter of the DISP parameter. There are four different subparameters that can be coded. These subparameters allow you to tell the system:

- The data set is to be created in the job step -- NEW.
- The data set existed before this job step -- OLD.
- The data set can be used by other concurrently executing jobs -- SHR.
- The data set is to be lengthened with additional output -- MOD.

At the begining of each job, the system determines by the status you specify in the DISP parameter whether a job is to have exclusive control of a data set. By specifying OLD, NEW, or MOD, you are requesting exclusive control of a data set for the duration of your job. In order to modify an existing data set, you should have exclusive control. If you plan for your job step or job to only read a data set and not to modify it, then you can request shared control of the data set.

The status of a data set that is defined and used in more than one step of a job is determined by the most restrictive status specified. The status that the system assigns to a data set from the information on the DD statement is not for the duration of the step but for the duration of the job. Therefore, in a three step job, if OLD is specified as the status in one step and SHR as the status in the other two steps, the status of the data set for all three steps would be OLD. OLD is a more restrictive status than SHR. This means that if exclusive control of a data set is requested anywhere in a job the data set cannot be shared. In order to share a data set, SHR must be specified every time the data set is defined in the job.

When you Specify NEW as the Data Set's Status

Specifying NEW as the first subparameter of the DISP parameter tells the system that the data set is to be created in the job step and may be used by the processing program to contain output data. If you omit the subparameter NEW, the system assumes the data set is to be created in the job step. (If you omit the subparameter NEW and specify a disposition or conditional disposition, you must code a comma to indicate the absence of NEW.) When the status of a data set is NEW, you must code on the DD statement all of the parameters necessary to define the data set.

Coding NEW guarantees exclusive control of the nontemporary data set name specified in the DSNAME parameter for the data set. Exclusive control of the data set name means that no other job that requests the data set, by that name, can be processed until the job with exclusive control terminates. This also means that the data set name itself is being exclusively controlled. If a request is made for the same data set name, the request will not be processed -- even though the request may refer to an entirely different physical data set.

When you Specify OLD as the Data Set's Status

Specifying OLD as the first subparameter of the DISP parameter tells the system that the data set existed before this job step.

Coding OLD guarantees exclusive control of the nontemporary data set name specified in the DSNAME parameter for the data set. Exclusive control of the data set name means that no other job that requests the data set, by that name, can be processed until the job with exclusive control terminates. This also means that the data set name itself is being exclusively controlled. If a request is made for the same data set name, the request will not be processed -- even though the request may refer to an entirely different physical data set.

When you Specify SHR as the Data Set's Status

Specifying SHR as the first subparameter of the DISP parameter tells the system that the data set resides on a direct access volume or tape volume and other jobs that are executing concurrently with this job step may simultaneously use (share) the data set. When SHR is specified, any job step that uses the data set should only read the data set.

You should have exclusive control of a data set in order to add or update records. If you plan to modify a data set, you should specify OLD or MOD in the DISP parameter. To protect other users of a shared data set, care should be exercised when specifying SHR if you plan to modify the data set. Several users can share a data set and write into it if exclusive control of the data set is acquired. For more information on sharing a data set and on gaining exclusive control of a data set when you have specified SHR, refer to the OS Data Management Services publication.

Caution should be observed when specifying SHR for IBM processor output data sets since no provision is made for acquiring exclusive control of the data sets prior to writing (e.g. SYSGO for the Assembler).

If you code DISP=(SHR,DELETE) the system assumes OLD instead of SHR. Once you specify SHR for a data set, every reference to that data set within the job must specify SHR or the data set can no longer be used by concurrently executing jobs.

When You Specify MOD as the Data Set's Status

Specifying MOD as the first subparameter of the DISP parameter tells the system that when the data set is opened for output, the read/write mechanism is to be positioned after the last record in the data set. MOD is specified when you want to add records to a data set with sequential, indexed sequential, or partitioned organization. MOD should not be specified for data sets with direct organization. When MOD is specified and the number of volumes required to lengthen the data set may exceed the number of units requested, specify a volume count in the VOLUME parameter. This ensures that the data set can be extended to new volumes.

When extending an old data set which at creation specified UNIT=group name, the additional units allocated for extending will not necessarily be of that same group. The same device type will be allocated.

When MOD is specified, the system first assumes the data set exists. If the volume information for the data set is supplied on the DD statement, in the system catalog, or passed with the data set from a previous step and the data set is not there, the system will issue an appropriate error message. Specifying MOD for a new sequential data set causes the read/write mechanism to be positioned after the last record in the data set each time it is opened for output.

Specifying MOD quarantees exclusive control of the nontemporary data set name specified in the DSNAME parameter for the data set. Exclusive control means that no other job that requests the data set, by that name, can be processed until the job with control terminates. This also means that the data set name itself is being exclusively controlled. If a request is made for the same data set name, the request will not be processed -- even though the request may refer to an entirely different physical data set.

If MOD is specified and the volume information exists for a multivolume data set, the first volumes will be mounted on the units allocated. However, if the data set is opened for output with a disposition of MOD, OPEN will start with the last volume, requiring demounting of the first volumes if more volume serial numbers were specified than units allocated. To avoid this inefficiecy, specify DEFER in the UNIT parameter, VOL=REF (for tape data sets only), or an explicit volume sequence number in the VOLUME parameter.

If the data set is opened for OUTPUT or OUTIN, the system will mount the last volume to search for EOF, unless a volume sequence number is specified on the DD statement. In that case, the volume sequence number is used to determine the volume to be mounted.

When you lengthen a data set that has standard labels, DCB information in the data control block must agree with the DCB information contained in the data set label. Conflicting DCB information, specifically conflicting block sizes, may make the data set unusable by later jobs. Therefore, do not code the DCB information contained in the data set label on the DD statement. If this DCB information is coded in the DCB macro instruction, be sure it agrees with the information contained in the data set label.

If you extend a data set that has fixed block standard (FBS) records and the last block was a truncated one, an end-of-data set condition occurs when the truncated block is encountered. If an attempt is made to read the data set backward on magnetic tape, processing is terminated immediately (with an end-of-data set condition) upon reading the truncated block.

Specifying a Disposition for the Data Set

The second subparameter of the DISP parameter tells the system what is to be done with the data set at the end of the job step. If you want the data set to assume the same attributes it had before the job, you need not code the second subparameter of the DISP parameter. However, if a conditional dispostion is specified, you must code a comma to indicate the absence of the second subparameter. When the second subparameter is not coded, data sets that existed before the job continue to exist and data sets that were created in the job step are deleted. If you create a nontemporary data set in the job and assign a disposition of PASS to it, the data set is deleted at termination of the job step that receives the passed data set and does not assign a disposition to it. (The passed nontemporary data set is deleted at job termination if the data set is never received by a later job step.)

The system ignores the disposition you have coded and automatically keeps existing data sets and deletes new data sets when the step is abnormally terminated before the step begins execution, e.g., primary direct access space cannot be obtained.

Sometimes the system does not perform disposition processing. The system does no disposition processing of data sets when:

- The job step is bypassed because of an error that is found during interpretation of control statements, e.g., a control statement containing errors is read.
- The job step is bypassed because a return code test is satisfied.
- The job step makes a nonspecific request for a tape volume and the data set is never opened. There is one exception: If you make a nonspecific request for a tape volume for a new generation data set and the data set is never opened, the system catalogs the data set with a volume serial number of blanks. Tape generation data sets that were never opened and that were cataloged may only be uncataloged by specifying the data set name parameter in the format of generation data group name and the relative generation number, or by using the IEHPROGM utility program.
- The job step requests that the mounting of a direct access volume be deferred and the data set is never opened.

Except for the cases mentioned above, the specified disposition is in effect for the data set if the job step terminates normally or abnormally and you have not specified a conditional disposition as the third subparameter of the DISP parameter.

There are five dispositions that can be specified for a data set. These dispositions allow you to:

- Delete a data set -- DELETE.
- Keep a data set -- KEEP.
- Pass a data set to a later job step -- PASS.
- Catalog a data set -- CATLG.
- Uncatalog a data set -- UNCATLG.

When you Specify DELETE as the Disposition

Specifying DELETE as the second subparameter of the DISP parameter tells the system that you want the data set's space on the volume released at the end of the job step. If the catalogued data set is a qualified data set, e.g., A.B.C., the system will automatically delete any index levels which become superfluous except the highest index level. If the data set resides on a tape volume, the tape is rewound and the volume is available for use by other data sets at the end of the job step. If the data set resides on a direct access volume, the system removes the volume table of contents entry associated with the data set and the data set's space is available for use by other data sets at the end of the job step. However, if the direct access data set's expiration date or retention period has not expired, the system does not delete the data set. You can use the IEHPROGM utility program to remove the volume table of contents entry.

If you are deleting a cataloged data set, the entry for the data set in the system catalog is also removed, provided the system obtained volume information for the data set from the catalog, i.e., the volume's serial number was not coded on the DD statement. If the system did not obtain volume information from the catalog, the data set is still deleted but its entry in the catalog remains. If an error is encountered while attempting to delete a data set, its entry in the catalog will not be removed. You may use the IEHPROGM utility program to delete an entry from the catalog.

If the catalogued data set is a qualified data set, e.g., A.B.C., the system will automatically delete any index levels which become superfluous except the highest index level.

When you Specify KEEP as the Disposition

Specifying KEEP as the second subparameter of the DISP parameter tells the system that you want the data set kept intact until a subsequent job step or job requests that the data set be deleted or until the expiration date is passed. (You can specify a retention period or expiration date in the LABEL parameter when the data set is created. If neither is coded in the LABEL parameter, a retention period of zero days is assumed by the system.)

When you Specify PASS as the Disposition

Specifying PASS as the second subparameter of the DISP parameter tells the system that the data set is to be passed after it is used in a job step. The system retains unit and volume information for a passed data set; when you refer to the data set in a DD statement of a subsequent job step, do not code the VOLUME parameter. A passed data set may be referred to in a later job step. You continue to code PASS each time the data set is referred to until the last time it is used in the job. At this time, you assign it a final disposition. If you do not assign the data set a final disposition, the system deletes the data set if it was created in the job and keeps the data set if it existed before the job. (See Appendix C for special consideration when passing ISAM data sets.)

When the data set is not in use, the volume that contains the passed data set remains mounted; therefore, you need not code RETAIN in the VOLUME parameter of a DD statement that specifies a disposition of PASS. If the system must remove the volume that contains the passed data set, it ensures through messages to the operator that the volume is remounted before the data set is used again.

When a subsequent job step wants to use the passed data set, you must include a DD statement for the step. On this DD statement, you must always code the DSNAME and DISP parameters.

- The DSNAME parameter identifies the data set. Either code the data set's name or make a backward reference to an earlier DD statement in the job that defines the data set.

- The DISP parameter specifies the data set's status and disposition. (If a later job step is to use this data set, specify a disposition of PASS; if this is the last job step that uses this data set, specify the data set's final disposition.)

The other parameters you might code are UNIT, LABEL, and DCB.

- Code the UNIT parameter if you want more than one device allocated to the data set.

- Code the LABEL parameter if you want to override the processing method specified in the OPEN macro instruction. See the IN and OUT subparameters of the LABEL parameter. (If you are processing a DOS unlabeled tape, code LABEL=(,LTM) if you want the system to test for and bypass a leading tapemark if it is encountered.)

- Code the DCB parameter or a backward reference to the DCB if the DCB information is not supplied by any other source.

If several data sets used in the job have the same name, you can only pass one of these data sets at a time. A job step must refer to a passed data set and assign a disposition of other than PASS to the data set before another data set with the same name can be passed. Only one reference to the data set name of a passed data set should be made within a single job step.

When you Specify CATLG as the Disposition

Specifying CATLG as the second subparameter of the DISP parameter tells the system to create a data set entry in the system catalog that points to this data set. The disposition CATLG also implies a disposition of KEEP. Once the data set is cataloged, you can retrieve the data set in later job steps and jobs by coding the DSNAME parameter and a status of other than NEW in the DISP parameter.

You can specify a disposition of CATLG for an already cataloged data set. This should be done when you are lengthening the data set with additional output (a status of MOD is coded) and the data set may exceed one volume. If the system obtained volume information for the data set from the catalog and you code DISP=(MOD,CATLG), the system updates the entry to include the volume serial numbers of any additional volumes. If there is more than one DD card within the same step reference which extends the same data set, lost data and an incorrect catalog entry may result.

If the data set's name is enclosed in apostrophes, the data set must not be assigned a disposition of CATLG. If the data set you want to catalog has a qualified name, e.g., A.B.C., the system will automatically create all the necessary index levels.

When you Specify UNCATLG as the Disposition

Specifying UNCATLG as the second subparameter of the DISP parameter tells the system that you want the data set's entry in the system catalog removed at the end of the job step; UNCATLG does not tell the system to delete the data set. Later jobs that use this data set must provide on the DD statement all of the parameters necessary to define the data set. If the data set you want to UNCATLG is a qualified data set, e.g., A.B.C., the system will automatically delete any index except the highest level index that becomes superfluous when the data set is removed from the catalog.

Specifying a Conditional Disposition for the Data Set

The third subparameter of the DISP parameter tells the system what is to be done with the data set if the step abnormally terminates. If you do not specify a conditional disposition and the step abnormally terminates, the system uses the disposition specified as the second subparameter of the DISP parameter to determine what is to be done with the data set. (There are a few exceptions and they are noted under "Specifying a Disposition for the Data Set.") If a passed data set has not been received and a job step abnormally terminates, the passed data set assumes the conditional disposition specified the last time it was passed. In this case, conditional disposition processing is done at job termination, not at step termination.

There are four conditional dispositions. When a job step abnormally terminates, these conditional dispositions allow you to:

- Delete a data set -- DELETE.
- Keep a data set -- KEEP.
- Catalog a data set -- CATLG.
- Uncatalog a data set -- UNCATLG.

When you Specify DELETE as the Conditional Disposition

Specifying DELETE as the third subparameter of the DISP parameter tells the system that if the step abnormally terminates you want the data set's space on the volume released. DELETE is the only valid conditional disposition that can be specified for a data set assigned a temporary name or no name. If the data set resides on a tape volume, the tape is rewound and the volume becomes available for use by other data sets at the end of the job step. If the data set resides on a direct access volume, the system removes the volume table of contents entry associated with the data set and the data set's space is available for use by other data sets at the end of the job step. However, if the direct access data set's expiration date or retention period has not expired, the system does not delete the data set. You can use the IEHPROGM utility program to remove the volume table of contents entry.

If the data set is cataloged, its entry in the system catalog is also removed, provided the system obtained volume information for the data set from the catalog, i.e., the volume's serial number was not coded on the DD statement. If the catalogued data set is a qualified data set, e.g., A.B.C., the system will automatically delete any index levels which become superfluous except the highest index level. If the system did not obtain volume information from the catalog, the data set is still deleted but its entry in the catalog remains. In this case, you may use the IEHPROGM utility program to delete the entry.

When you Specify KEEP as the Conditional Disposition

Specifying KEEP as the third subparameter of the DISP parameter tells the system that if the step abnormally terminates you want the data set kept intact until a subsequent job requests that the data set be deleted or until the expiration date has passed. (You can specify a retention period or expiration date in the LABEL parameter when the data set is created. If neither is coded in the LABEL parameter, a retention period of zero days is assumed by the system.)

Note: A scratch volume will be rewound, unloaded, and a KEEP message issued to the operator during abnormal termination of a job step when: (1) a temporary data set written on the scratch volume has been assigned a nontemporary name, and (2) a conditional disposition of KEEP has been assigned to the data set.

When you Specify CATLG as the Conditional Disposition

Specifying CATLG as the third subparameter of the DISP parameter tells the system that if the step abnormally terminates you want the system to create an entry in the system catalog that points to this data set. The conditional disposition of CATLG also implies a conditional disposition of KEEP. Once the data set is cataloged, you can retrieve the data set in later job steps and jobs by coding the DSNAME parameter and a status of other than NEW in the DISP parameter.

If the data set's name is enclosed in apostrophes, the data set must not be assigned a conditional disposition of CATLG. If the data set has a qualified name, e.g., A.B.C., the system will automatically create all the necessary index levels.

When you Specify UNCATLG as the Conditional Disposition

Specifying UNCATLG as the third subparameter of the DISP parameter tells the system that if the step abnormally terminates you want the data set's entry in the system catalog removed, UNCATLG does not tell the system to delete the data set. Later jobs that use this data set must provide on the DD statement all of the parameters necessary to define the data set. if the data set has a qualified name, e.g., A.B.C., the system will automatically delete any index except the highest level index that becomes superfluous when the data set is removed from the catalog.

The system performs disposition processing of data sets at step termination. This processing is based on whether the step terminated normally or abnormally, the data set's status, the requested disposition, and the conditional disposition. Figure 23 shows the disposition processing performed by the system based on these factors.

Status	Requested Disposition	Conditional Disposition	Action Taken at Normal End of Step[1]	Action Taken at Abnormal End of Step[1], when Step Fails Due to:			Action Taken at End of Job
				No Allocation Performed	Job Cancelled After Data Set Allocation or Program Check	After Data Set Allocation, Another Allocation Cannot be Performed	
NEW or MOD[2]	none	none	deleted	deleted	deleted	deleted	
	KEEP	none	kept	deleted	kept	deleted	
	DELETE	none	deleted	deleted	deleted	deleted	
	CATLG	none	cataloged	deleted	cataloged	deleted	
	PASS	none	passed	deleted	passed	passed	deleted
	PASS	any except UNCATLG[3]	passed	deleted	passed	passed	conditional disposition
	any except PASS	KEEP	requested disposition	deleted	kept	kept	
	any except PASS	DELETE	requested disposition	deleted	deleted	deleted	
	any except PASS	CATLG	requested disposition	deleted	cataloged	cataloged	
OLD or MOD or SHR	none	none	kept	kept	kept	kept	
	KEEP	none	kept	kept	kept	kept	
	DELETE	none	deleted	kept	deleted	kept	
	CATLG	none	cataloged	kept	cataloged	kept	
	UNCATLG	none	uncataloged	kept	uncataloged	kept	
	PASS	none	passed	kept	passed	passed	kept
	PASS	any	passed	kept	passed	passed	conditional disposition
	any except PASS	KEEP	requested disposition	kept	kept	kept	
	any except PASS	DELETE	requested disposition	kept	deleted	deleted	
	any except PASS	CATLG	requested disposition	kept	cataloged	cataloged	
	any except PASS	UNCATLG	requested disposition	kept	uncataloged	uncataloged	

Footnotes:

1 See list of exceptions in right-hand column.

2 For MOD, a data set is considered to be a new data set if volume information is not available to the system.

3 A conditional disposition other than DELETE is invalid for a data set that is assigned a temporary name or no name. The system assumes DELETE.

List of Exceptions:

1. When a nontemporary data set is passed and the receiving step does not assign it a disposition, the system will, upon termination of this step, do one of two things. If the data set was new when it was initially passed, it will be deleted. If the data set was old when initially passed, it will be kept. Temporary data sets are deleted.
2. If a job step makes a nonspecific request for a tape volume with the disposition of PASS and the data set is not opened in the step in which it is created, the job will ABEND.
3. If a job step requests that the mounting of a direct access volume be deferred and the data set is never opened, no disposition processing is performed.
4. If automatic step restart is to occur, all data sets in the restart step with a status of OLD or MOD, and all data sets being passed to steps following the restart step, are kept. All data sets in the restart step with a status of NEW are deleted.
5. If automatic checkpoint restart is to occur, all data sets currently in use by the job are kept.
6. When dedicated data sets are used in a job step, any disposition assigned to them is internally changed to PASS or KEEP to prevent deletion of the dedicated data sets.

Figure 22. Disposition Processing Chart

Examples of the DISP Parameter

```
1.  //DD       DD   DSNAME=D99.GROUP.SIX,UNIT=2311,VOLUME=SER=111111, X
    //             DISP=(NEW,CATLG,DELETE),SPACE=(TRK,(5,1))
```

This DD statement defines a new data set and requests the system to create an index entry in the system catalog that points to this data set if the step terminates normally. It also requests the system to delete the data set, instead of cataloging it, if the step abnormally terminates. The system will automatically create the indexes in the catalog for D99 and GROUP.

```
2.  //DD2      DD   DSNAME=FIX,UNIT=2400-1,VOLUME=SER=44889,          X
    //             DISP=(OLD,,DELETE)
```

This DD statement defines an existing data set and implies that the data set is to be kept if the step terminates normally. (For an existing data set, the system assumes it is to keep the data set if no disposition is specified.) The statement requests the system to delete the data set if the step abnormally terminates.

```
3.  //STEP1    EXEC PGM=FILL
    //DD1      DD   DSNAME=SWITCH.LEVEL18.GROUP12,UNIT=2311,          X
    //             VOLUME=SER=LOCAT3,SPACE=(TRK,(80,15)),DISP=(,PASS)
    //STEP2    EXEC PGM=CHAR
    //DD2      DD   DSNAME=XTRA,DISP=OLD
    //DD3      DD   DSNAME=*.STEP1.DD1,DISP=(OLD,PASS,DELETE)
    //STEP3    EXEC PGM=TERM
    //DD4      DD   DSNAME=*.STEP2.DD3,DISP=(OLD,CATLG,DELETE)
```

The DD statement named DD1 in STEP1 defines a new data set and requests that the data set be passed. If STEP1 abnormally terminates, the data set is deleted since it is a new data set and a conditional disposition was not specified. The DD statement named DD3 in STEP2 receives the passed data set and requests that the data set be passed. If STEP2 abnormally terminates, the data set is deleted because of the conditional disposition of DELETE. The DD statement named DD4 in STEP3 receives the passed data set and requests that the data set be cataloged at the end of the step. If STEP3 abnormally terminates, the data set is deleted because of the conditional disposition of DELETE.

The DLM Parameter

```
DLM=delimiter
```

delimiter
the characters that will indicate the end of a group of data in the input stream.

Rules for Coding

1. The delimiter can be any combination of two characters.

2. If the delimiter contains any special characters, you must enclose the delimiter in apostrophes (5-8 punch).

3. If you include an ampersand or an apostrophe in the delimiter, you must code each ampersand or apostrophe as two consecutive ampersands or apostrophes.

4. The DLM parameter has meaning only on statements defining data in the input stream (DD * and DD DATA statements).

5. If you do code the DLM parameter on a DD * or DD DATA statement, the characters you assign as delimiter override the /* delimiter on the DD * or DD DATA statements. You can then terminate the data with the characters assigned in the DLM parameter.

WHAT THE DLM PARAMETER DOES

The DLM parameter allows you to use a delimiter other than /* to terminate a group of data defined in the input stream. By assigning a different delimiter in the DLM parameter, you can include a delimiter statement with the delimiter /* as data in the input stream.

You code the DLM parameter on a DD * or DD DATA statement. The system recognizes the character you assign in the DLM parameter as the delimiter for the data in the input stream. The new delimiter has meaning only for that group of data defined by the DD statement containing the DLM parameter. If the system encounters an error on the DD statement before the DLM parameter, it will not recognize the value assigned as a delimiter.

The delimiter you assign is coded on the delimiter statement in the same way /* is coded. For detailed information, see the section on the delimiter statement.

Examples of the DLM Parameter

1.
```
//DD1     DD   *,DLM=AA
            .
            .
  Data
AA
```

The DLM parameter assigns the characters AA as the valid delimiter for the data defined in the input stream by DD1.

```
2.  //DD2      DD   DATA,DCB=BUFNO=2,DLM='&&&&'
                .
                .
      data
    &&
```

The DLM parameter assigns the characters && as the valid delimiter for the data defined in the input stream by DD2. Since an ampersand is a special character, it must be enclosed in apostrophes in the DLM parameter. Each ampersand must also be written as two consecutive ampersands in the DLM parameter.

The DSNAME Parameter

```
{DSNAME}   =   (dsname
{DSN   }       dsname(member name)
               dsname(generation number)
               dsname(area name)
               &&dsname
               &&dsname(member name)
               &&dsname(area name)
               *.ddname
               *.stepname.ddname
               *.stepname.procstepname.ddname)
```

dsname
: identifies a data set name.

dsname(member name)
: identifies a nontemporary partitioned data set name and the name of a member within that data set.

dsname(generation number)
: identifies a generation data group by its name and a generation data set by its generation number (a zero or signed integer.)

dsname(area name)
: identifies a nontemporary indexed sequential data set name and an area of that data set (INDEX, PRIME, or OVFLOW.)

&&dsname
: specifies the name you want assigned to a temporary data set.

&&dsname(member name)
: specifies the name you want assigned to a temporary partitioned data set and to a member within that data set.

&&dsname(area name)
: specifies the name you want assigned to a temporary indexed sequential data set and identifies an area of that data set (INDEX, PRIME, or OVFLOW.)

*.ddname
: specifies that the data set name is to be copied from the named DD statement, which is an earlier DD statement in the job step.

*.stepname.ddname
: specifies that the data set name is to be copied from an earlier DD statement named ddname, which appears in an earlier step named stepname in the same job.

*.stepname.procstepname.ddname
: specifies that the data set name is to be copied from an earlier DD statement in a cataloged procedure. Stepname is the name of the job step that calls the procedure, procstepname is the name of the procedure step that includes the named DD statement, and ddname is the name of the DD statement that contains the data set name.

Rules for Coding

1. An unqualified data set name may consist of 1 to 8 characters. The first character must be an alphabetic or national (@,$,#) character; the remaining characters can be any alphameric or national characters, a hyphen, or a plus zero (12-0 punch). A temporary data set name can consist of 1 through 8 characters, preceded by two

ampersands; the first character following the ampersands must be an alphabetic or national character.

2. A qualified name may consist of up to 44 characters including periods. For each eight characters or less there must be a period, and the character following a period must be an alphabetic or national (@,$,#) character.

3. You need not code the DSNAME parameter if the data set is created and deleted in the job, i.e., if the data set is temporary.

4. The DSNAME and DDNAME parameters are mutually exclusive parameters; therefore, when the DDNAME parameter is coded, do not code the DSNAME parameter. The statement to which the DDNAME parameter refers can contain the DSNAME parameter.

IDENTIFYING THE DATA SET

When you create a data set, you use the DSNAME parameter to assign a name to the data set. The data set name is part of the information stored with the data set on a volume. Later, when another job step or job wants to use the data set, it identifies the data set name in the DSNAME parameter; the system uses the data set name to locate the data set on the volume.

How you code the DSNAME parameter depends on the type of data set and whether the data set is nontemporary or temporary.

Creating or Retrieving a Nontemporary Data Set

If the data set is nontemporary, you can identify:

- A nontemporary data set by coding DSNAME=dsname.
- A member of a nontemporary partitioned data set by coding DSNAME=dsname(member name).
- A generation of a nontemporary generation data group by coding DSNAME=dsname(number).
- An area of a nontemporary indexed sequential data set by coding DSNAME=dsname(area name).

NONTEMPORARY DATA SET

When a nontemporary data set is created, it is assigned a name in the DSNAME parameter and is assigned a disposition of KEEP or CATLG. (A data set assigned a disposition of KEEP may be assigned a disposition of CATLG by a later job step or job.) The name you assign to a nontemporary data set must be specified in the DSNAME parameter by all other steps and jobs that want to use the data set.

A nontemporary data set name can be either an unqualified or qualified name. An unqualified data set name consists of 1 through 8 characters. The first character must be an alphabetic or national (@,#,$) character; the remaining characters can be any alphameric or national characters, a hyphen, or a plus zero (12-0 punch).

A qualified data set name consists of 1 through 44 characters (including periods), except when the qualified name identifies a generation data group. In this case, the data set name may consist of only 1 through 35 characters (including periods). For each eight

characters or less there must be a period, and the first character of the name and the character following a period must be an alphabetic or national (@,#,$) character.

If you assign a qualified name to a data set that is to be cataloged, the system will generate all the nesessary index levels in the catalog.

When you request a data set that is cataloged on a control volume other than the system catalog, the system attempts to mount this control volume if it is not already mounted. After the system obtains the pointer to this data set, the control volume may then be demounted by the system if the unit on which it was mounted is required by another volume. If you plan to delete, uncatalog, or recatalog the data set, the volume must be mounted during disposition processing (at the end of the job step) in order for the pointer to be deleted or revised. You can ensure that the volume remains mounted by requesting the operator to issue a MOUNT command for this volume before the job step is initiated. If you do not use the MOUNT command to mount the volume and if the volume is not mounted during disposition processing, then, after the job has terminated, use the IEHPROGM utility program to delete or revise the pointer in the control volume. (In order for the system to mount a control volume, the control volume must be logically connected to the system catalog. This is done using the CONNECT function of the IEHPROGM utility program, which is described in the Utilities publication.)

MEMBERS OF A PARTITIONED DATA SET

A partitioned data set consists of independent groups of sequential records, each identified by a member name in a directory. When you want to add a member to a partitioned data set or retrieve a member, you specify the partitioned data set name and follow it with the member name. The member name is enclosed in parentheses and consists of 1 to 8 characters. The first character must be an alphabetic or national (@,$,#) character; the remaining characters can be any alphameric or national characters.

GENERATIONS OF A GENERATION DATA SET

A generation data group is a collection of chronologically related data sets that can be referred to by the same data set name. When you want to add a generation to a generation data group or retrieve a generation, you specify the generation data group name and follow it with the generation number. The generation number is enclosed in parentheses and the number is a zero or a signed integer. A zero represents the highest generation number present in the catalog at the time the current job was started; a negative integer (e.g.,-1) represents an older generation of the group; a positive integer (e.g.,+1) represents a new generation that is to be or has been created in this job and was not present in the catalog at job initiation.

To retrieve all generations of a generation data group (up to 255 generations), code only the group name in the DSNAME parameter and the DISP parameter.

A complete discussion of creating and retrieving generation data sets is contained in "Appendix D: Creating and Retrieving Generation Data Sets" in this publication.

AREAS OF AN INDEXED SEQUENTIAL DATA SET

The areas used for an indexed sequential data set are the index, prime, and overflow areas. When you are creating the data set and define any of these areas on a DD statement, you must identify the data set name and follow it with the area name you are defining. The area name is enclosed in parentheses and is either PRIME, INDEX, or OVFLOW. If you are using only one DD statement to define the entire data set, code DSNAME=dsname or DSNAME=dsname(PRIME). When you retrieve a data set, the term PRIME, INDEX, or OVFLOW is optional. For detailed information on how to create and retrieve indexed sequential data sets, refer to "Appendix C: Creating and Retrieving Indexed Sequential Data Sets" in this publication.

Creating or Retrieving a Temporary Data Set

If the data set is temporary, you can identify:

- A temporary data set by coding DSNAME=&&dsname.
- A member of a temporary partitioned data set by coding DSNAME=&&dsname(member name).
- An area of a temporary indexed sequential data set by coding DSNAME=&&dsname(area name).

TEMPORARY DATA SETS

Any data set that is created and deleted within the same job is a temporary data set. A DD statement that defines a temporary data set need not include the DSNAME parameter; the system generates one for you.

If you do include the DSNAME parameter, the temporary data set name can consist of 1 through 8 characters and is preceded by two ampersands (&&). The character following the ampersands must be an alphabetic or national (@,#,$) character; the remaining characters can be any alphameric or national characters. (A temporary data set name that is preceded by only one ampersand is treated as a temporary data set name as long as no value is assigned to it either on the EXEC statement for this job step when it calls a procedure, or on a PROC statement within the procedure. If a value is assigned to it by one of these means, it is treated as a symbolic parameter. Symbolic parameters are discussed in Appendix A.)

The system generates a qualified name for the temporary data set with the following format:

SYSyyddd.Ttttttt.xyzzz.jobname. {&name / unique no.}

The characters are identified as follows:

SYSyyddd	SYS - a constant followed by the date yy - the two-digit year ddd - the three-digit day
Ttttttt	T - a constant followed by a time stamp ttttttt - seven-digit time stamp

xyzzz	x - indicates the type of data set: R - sysin S - sysout I - pre-spooled sysin y - indicates the system configuration: F - MFT V - MVT zzz - three numeric characters assigned to make the name of the data set unique.
jobname	the name of the job for which the data set was created
&name if specified by user or a unique no.	the name you assign to the data set in the DSNAME parameter a character string beginning with R or S and ending with a seven-digit unique number: R - sysout and user-defined temporary data sets S - sysin and pre-spooled sysin data sets

The date and time are constants established for the specific reader creating the temporary data set name. They are initialized when the reader is started. Every temporary data set name created by a specific reader during its operation will have the same date and time.

If you attempt to keep or catalog a temporary data set (you specify a disposition of KEEP or CATLG in the DISP parameter), the system changes the disposition to PASS and the data set is deleted at job termination. However, this change is not made for a data set on a tape volume when the following conditions exist: (1) the data set is new; (2) the data set is not assigned a name; and (3) DEFER is specified in the UNIT parameter. The data set is deleted at job termination, but the system tells the operator to keep the volume on which the data set resided during the job.

MEMBERS OF A TEMPORARY PARTITIONED DATA SET

When you want to add a member to a temporary partitioned data set or retrieve a member during the job, you specify the partitioned data set's temporary name and follow it with the member name. The member name is enclosed in parentheses and consists of 1 to 8 characters. The first character must be an alphabetic or national (@,$,#) character; the remaining characters can be any alphameric or national characters.

AREAS OF A TEMPORARY INDEXED SEQUENTIAL DATA SET

The areas used for an indexed sequential data set are the index, prime, and overflow areas. When you are creating a temporary indexed sequential data set and define any of these areas on a DD statement, you must identify the data set's temporary name and follow it with the area name you are defining. The area name is enclosed in parentheses and is either PRIME, INDEX, or OVFLOW. If you are using only one DD statement to define the entire temporary data set, code DSNAME=&&dsname or DSNAME=&&dsname(PRIME). If you want to retrieve a data set, the term PRIME, INDEX or OVFLOW is optional. For information on how to create and retrieve indexed sequential data sets, refer to "Appendix C: Creating and Retrieving Indexed Sequential Data Sets" in this publication.

USING A DEDICATED DATA SET

If your installation provides dedicated data sets in a system with MVT, you can use these data sets to contain your data instead of creating your own temporary data sets. The use of dedicated data sets eliminates some of the time required to schedule a job step since the data sets are already allocated.

To use a dedicated data set, code DSNAME=&&name or DSNAME=&name on a DD statement, along with all other parameters required to define your temporary data set, e.g., UNIT, SPACE, DCB. Replace the term "name" with the ddname of the DD statement in the initiator cataloged procedure that defines the dedicated data set you want to use. If the system cannot assign you this dedicated data set, the parameters coded on your DD statement are used to create a temporary data set. (For detailed information on dedicated data sets, refer to the MVT Guide.)

Copying the Data Set Name From an Earlier DD Statement

The name of a data set that is used several times in a job, whether specified in the DSNAME parameter or assigned by the system, can be copied after its first use in the job. This allows you to easily change data sets from job to job and eliminates your having to assign names to temporary data sets. To copy a data set name, refer to an earlier DD statement that identifies the data set. When the earlier DD statement is contained in an earlier job step, you code DSNAME=*.stepname.ddname; when the earlier DD statement is contained in the same job step, you code DSNAME=*.ddname; when the earlier DD statement is contained in a cataloged procedure step called by an earlier job step, you code DSNAME=*.stepname.procstepname.ddname.

Note: If you copy the name for a new data set from an earlier data set that was assigned a disposition of DELETE, the new data set will be temporary. You should assign a disposition of PASS to the new data set. If you do not specify a disposition, or specify a disposition other than PASS, the system assumes PASS.

Specifying the DSNAME Parameter in Apostrophes

Sometimes, it may be necessary or desirable to specify a data set name that contains special characters. If the name contains special characters, you must enclose the name in apostrophes (5-8 punch), e.g., DSNAME='DAT+5'. If one of the special characters is an apostrophe, you must identify it by coding two consecutive apostrophes (two 5-8 punches) in its place, e.g., DSNAME='DAY''SEND'. A data set name enclosed in apostrophes can consist of 1 through 44 characters.

There are cases when your data set name must contain required special characters, which tell the system something about the data set (e.g., && in DSNAME=&&name are required special characters that tell the system that this is a temporary data set). In these cases, the data set name must not be enclosed in apostrophes because the system will not recognize the required special characters as having any special significance. The following data set names contain special characters that tell the system something about the data set and, therefore, cannot be enclosed in apostrophes:

- DSNAME=name(member name)
- DSNAME=name(area name)
- DSNAME=name(generation number)
- DSNAME=&&name
- DSNAME=*.stepname.ddname

Keep the following rules in mind:

1. If the data set is to be cataloged, the data set name cannot be enclosed in apostrophes.

2. If the data set name begins with a blank character, the data set is assigned a temporary data set name by the system.

3. If the data set name ends with a blank character, the blank is ignored.

4. If the only special character is a period or a hyphen, you need not enclose the data set name in apostrophes.

5. If retreiving a data set, unit and volume information should be supplied in the DD statement or should be received as a passed data set.

Examples of the DSNAME Parameter

1.
```
//DD1      DD   DSNAME=ALPHA,DISP=(,KEEP),                    X
//             UNIT=2400,VOLUME=SER=389984
```

This DD statement defines a new data set whose name is ALPHA. Later job steps or jobs may retrieve this data set by supplying the data set name in the DSNAME parameter, unit information in the UNIT parameter, and volume information in the VOLUME parameter.

2.
```
//DD2      DD   DSNAME=PDS(PROG12),DISP=(OLD,KEEP),UNIT=2311,    X
//             VOLUME=SER=882234
```

This DD statement retrieves a member of a partitioned data set named PDS.

3.
```
//DD3      DD   DSNAME=&&WORK,UNIT=2400
```

This DD statement defines a temporary data set. Since the data set is deleted at the end of the job step, the DSNAME parameter could be omitted.

4.
```
//STEP1    EXEC PGM=CREATE
//DD4      DD   DSNAME=&&ISDATA(PRIME),DISP=(,PASS),UNIT=(2311,2), X
//             SPACE=(CYL,(10,,2),,CONTIG),VOLUME=SER=(33489,33490)
//STEP2    EXEC PGM=OPER
//DD5      DD   DSNAME=*.STEP1.DD4,DISP=(OLD,DELETE)
```

The DD statement named DD4 in STEP1 defines a temporary indexed sequential data set whose name is ISDATA. This DD statement is used to define all of the areas of an indexed sequential data set. The DD statement named DD5 in STEP2 retrieves the data set by referring to the earlier DD statement that defines the data set. Since the temporary data set is passed when it is defined in STEP1, STEP2 can retrieve the data set.

The FCB Parameter

```
                    FCB=(image-id [,ALIGN  ] )
                                  [,VERIFY ]
```

image-id
    the code that identifies the image to be loaded into the forms control

,ALIGN
    requests the operator to check the alignment of the printer forms before the data set is printed.

,VERIFY
    requests the operator to visually verify the image displayed on the printer as the desired one. The operator is also given an opportunity to align the printer forms.

Rules for Coding

1. The image-id can be 1 to 4 characters in length.

2. The FCB parameter is ignored if the data set is not written to a 3211 printer.

3. The FCB and DDNAME parameters and the DCB subparameters RKP, CYLOFL, and INTVL are mutually exclusive parameters; therefore, if you code the DDNAME parameter or one of the DCB subparameters RKP, CYLOFL, or INTVL, do not code the FCB parameter.

4. If you do not code ALIGN or VERIFY, you need not enclose the image-id in parentheses.

Image Identifier

The image-id is the code that identifies the image to be loaded into the forms control buffer (FCB). It is retrieved from SYS1.IMAGELIB or defined in the user's program through the exit list facility of the DCB macro instruction. IBM provides two standard FCB images, STD1 and STD2.

STD1 specifies that 6 lines per inch are to be printed on an 8.5 inch form. STD2 specifies that 6 lines per inch are to be printed on an 11 inch form. The installation can provide additional user-designed images.

If you omit the FCB parameter and the data set is written to a 3211 printer, the default image is used if it is currently in the buffer. Otherwise, the operator will be requested to specify an image.

REQUESTING ALIGNMENT OF FORMS

If you want to request that the operator check the alignment of the printer forms before the data set is printed, code ALIGN as the second subparameter of the FCB parameter.

REQUESTING OPERATOR VERIFICATION

By specifying VERIFY, you can request that the operator visually verify that the image displayed on the printer is the desired one. Specifying VERIFY also gives the operator an opportunity to align the forms.

Examples of the FCB Parameter

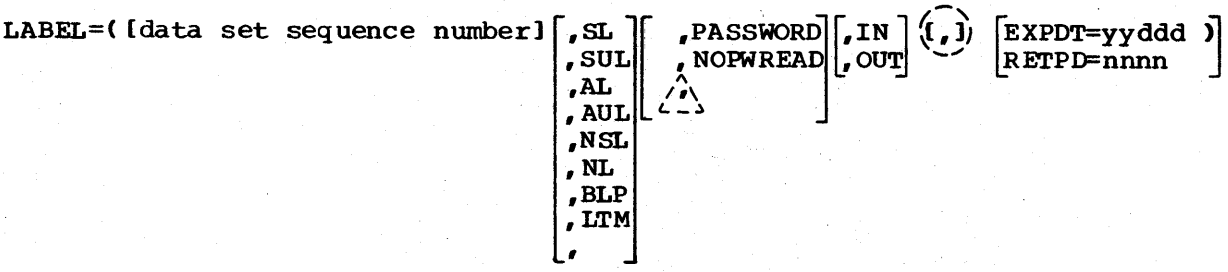
1. `//DD1 DD UNIT=3211,FCB=(IMG1,VERIFY)`

   This DD statement defines the output data set that is to be written to a 3211 printer. The FCB parameter requests that the data set be written using the control information corresponding to the forms control image with the code IMG1. Since VERIFY is coded, the forms control image is displayed on the printer before the data set is printed and the operator is asked to align the printer forms.

2. `//DD2 DD SYSOUT=A,FCB=IMG2`

   This DD statement defines an output data set that is to be written to the device that corresponds with class A. The FCB parameter is ignored if the device is not a 3211 printer.

The LABEL Parameter



data set sequence number
specifies the relative position of a data set on a tape volume.

,SL
specifies that the data set has IBM standard labels.

,SUL
specifies that the data set has both IBM standard and user labels.

,AL
specifies that the data set has American National Standard labels.[1]

,AUL
specifies that the data set has both American National Standard labels and American National Standard user labels.[1]

,NSL
specifies that the tape data set has nonstandard labels.

,NL
specifies that the tape data set has no labels.

,BLP
specifies that the system is not to perform label processing for the tape data set.

,LTM
specifies that the system is to check for and bypass a leading tapemark on a DOS unlabeled tape.

⚠
specifies that the data set has standard labels and another subparameter follows.

,PASSWORD
specifies that the new data set cannot be used by another job step or job unless the operator can supply the system with the correct password, i.e., the data set cannot be read, changed, extended, or deleted.

,NOPWREAD
specifies that the data set can be read without the password, but the operator must give the password before the data set can be changed, extended, or deleted.

---

[1]The DCB subparameter OPTCD will be treated as if DCB=OPTCD=Q was specified on the DD statement. If you specify an OPTCD on the DD statement, it will be treated as if Q was specified in addition to any other characters you code.

,
specifies that another subparameter follows and, for a data set, the data set is not to be password protected.

,IN
specifies that the data set is to be processed for input only.

,OUT
specifies that the data set is to be processed for output only.

(,)
specifies that either the RETPD or EXPDT subparameter follows and one or more subparameters precede it.

EXPDT=yyddd
specifies the date when the data set can be deleted or overwritten by another data set. Assign a 2-digit year number and a 3-digit day number.

RETPD=nnnn
specifies the length of time in days that the data set must be kept. Assign the number of days that must pass before the data set can be deleted or overwritten by another data set.

Rules for Coding

1. All the subparameters except the last subparameter in the LABEL parameter are positional subparameters. Therefore, if you want to omit a subparameter, you must indicate its absence with a comma.

2. If the only subparameter you want to specify is the data set sequence number, RETPD or EXPDT, you can omit the parentheses and commas and code LABEL=data set sequence number, LABEL=RETPD=nnnn, or LABEL=EXPDT=yyddd.

3. If the data set has IBM standard labels, you can omit the subparameter SL.

4. When you are defining a data set that resides or will reside on a direct access volume, only SUL or SL can be specified as the second subparameter.

5. If you are processing ASCII data on unlabeled (NL) tapes, you must code OPTCD=Q in your DCB macro instruction or in the DCB parameter on the DD statement.

6. Do not code LTM for magnetic tapes with labels. If you do code LTM for a labeled tape, the system will reject the tape.

7. The LABEL, DDNAME, and SYSOUT parameters are mutually exclusive parameters; therefore, if DDNAME or SYSOUT is coded, do not code the LABEL parameter.

DATA SET LABELS

Labels are used by the operating system to identify volumes and the data sets they contain, and to store data set attributes. Data sets residing on magnetic tape volumes usually have data set labels. If data set labels are present, they precede each data set on the volume. Data sets residing on direct access volumes always have data set labels. These data set labels are contained in the volume table of contents at the beginning of the direct access volume.

A data set label may be a standard or nonstandard label. Standard labels can be processed by the system; nonstandard labels must be processed by nonstandard label processing routines, which the installation includes in the system. Data sets on direct access volumes must have standard labels. Data sets on tape volumes usually have standard labels, but can have nonstandard labels or no labels.

Tape label definitions and associated tape label processing are included in the Tape Labels publication. Direct access label definitions and associated direct access label processing are described in "Appendix A: Direct Access Labels" in the Data Management Services publication.

When to Code the LABEL Parameter

The LABEL parameter must be coded if:

- You are processing a tape data set that is not the first data set on the reel; in this case, you must indicate the data set sequence number.
- The data set labels are not IBM standard labels; you must indicate the label type.
- You want to specify what type of labels a data set is to have when it is written on a scratch volume; you must indicate the label type.
- The data set is to be password protected; you must specify PASSWORD when you create the data set.
- The data set is to be processed only for input or output and this conflicts with the processing method indicated in the OPEN macro instruction; you must specify IN, for input, or OUT, for output.
- The data set is to be kept for some period of time; you must indicate a retention period (RETPD) or expiration date (EXPDT).
- You are processing a DOS unlabeled tape and want the system to check for and bypass a leading tapemark on the tape.

THE DATA SET SEQUENCE NUMBER SUBPARAMETER

When you want to place a data set on a tape volume that already contains one or more data sets, you must specify where the data set is to be placed, i.e., the data set is to be the second, third, fourth, etc., data set on the volume. The data set sequence number causes the tape to be positioned properly so that the data set can be written on the tape or retrieved.

The data set sequence number subparameter is a positional subparameter and is the first subparameter that can be coded. The data set sequence number is a 1- to 4-digit number. The system assumes 1, i.e., this is the first data set on the reel, if you omit this subparameter or if you code 0, unless the data set is a passed or cataloged data set. If a data set is cataloged, the system obtains the data set sequence number from the catalog; for a passed data set, the data set sequence number is obtained from the passing step.

When you request the system to bypass label processing (BLP is coded as the label type in the LABEL parameter) and the tape volume contains labels, the system treats anything between tapemarks as a data set. Therefore, in order for the tape with labels to be positioned properly, the data set sequence number must reflect all labels and data sets that precede the desired set. Section I of the Tape Labels publication illustrates where tapemarks appear.

THE LABEL TYPE SUBPARAMETER

The label type ubparameter tells the system what type of label is associated with the data set. The label type subparameter is a positional subparameter and must be coded second, after the data set sequence number subparameter. You can omit this subparameter if the data set has IBM standard labels.

The label type subparameter is specified as:

- SL -- if the data set has IBM standard labels.
- SUL -- if the data set has both IBM standard and user labels.
- AL -- if the data set has American National Standard labels.[1]
- AUL -- if the data set has American National Standard labels and American National Standard user.[1]
- NSL -- if the data set has nonstandard labels.
- NL -- if the data set has no labels.
- BLP -- if you want label processing bypassed.
- LTM -- if you want the system to check for and bypass a leading tapemark.

SL or SUL is the only label type that can be specified for data sets that reside on direct access volumes.

When SL or SUL is specified, or the label type subparameter is omitted and the data set has IBM standard labels, the system can ensure that the correct tape or direct access volume is mounted. When you specify NSL, installation-provided nonstandard label processing routines must ensure that the correct tape volume is mounted. When you specify NL, BLP or LTM, the operator must ensure that the correct tape volume is mounted. If you specify NL, the data set must not have standard labels. If you specify LTM, the tape must not have labels. When you specify AL or AUL, the system ensures that the correct American National Standard labeled tape is mounted.

For cataloged data sets, label type information is not kept. Therefore, any time you refer to a cataloged data set that has other than standard labels, you must code the LABEL parameter and specify the label type.

BLP is not a label type, but a request to the system to bypass label processing. This specification allows you to use a blank tape or overwrite a seven-track tape that differs from your current parity or density specifications. Bypass label processing is an option of the operating system, specified as a PARM field value in the reader cataloged procedure. If the option is not selected and you have coded BLP, the system assumes NL.

---

[1]The DCB subparameter OPTCD will be treated as if DCB=OPTCD=Q was specified on the DD statement. If you specify an OPTCD on the DD statement, it will be treated as if Q was specified in addition to any other characters you code.

Note for BLP: When you request the system to bypass label processing and the tape volume has labels, the system treats anything between tapemarks as a data set. Therefore, in order for a tape with labels to be positioned properly, the data set sequence number subparameter of the LABEL parameter must be coded and the subparameter must reflect all labels and data sets that precede the desired data set. In order to process a multi-volume data set using bypass label processing, it is necessary to treat each volume as a separate data set and concatenate them. In addition, if you change the label attributes in a subsequent step or job, you must ensure that the system demounts the tape before use. Section I of the Tape Labels publication illustrates where tapemarks appear.

LTM is not a label type but a request to the system to check for and bypass a leading tapemark on the tape. In some instances, a tapemark will precede the first data set on unlabeled tapes created by the Disk Operating System (DOS). Unlabeled tapes created by the System/360 Operating System do not contain leading tapemarks. The LTM subparameter allows you to use DOS unlabeled tapes with the System/360 Operating System without making modifications. You can also specify LTM for a multivolume data set on DOS tapes; the system will check for and bypass a leading tapemark on each volume. Do not code LTM with labeled tapes; if you specify LTM for a tape with labels, the system will reject the tape.

The label type subparameter can also be specified when you make a nonspecific volume request for a tape volume (i.e., no volume serial numbers are specified on the DD statement) and you want the data set to have a certain type of labels. If the volume that is mounted does not have the corresponding label type you desire, you may be able to change the label type.

When you specify NL or NSL and the operator mounts a tape volume that contains standard labels, you may use the volume provided: (1) the expiration date of the existing data set on the volume has passed; (2) the existing data set on the volume is not password protected; (3) you make a nonspecific volume request; and (4) the file sequence number is less than two. All of these conditions must be met. If they are not, the system requests the operator to mount another tape volume.

If you specify SL and make a nonspecific volume request, but the operator mounts a tape volume that contains other than IBM standard labels, the system requests the operator to identify the volume serial number and the volume's new owner before the IBM standard labels are written. If the tape volume has American National Standard labels, the system asks the operator for permission to destroy the label. If you specify SL and make a specific volume request, but the volume that is mounted does not contain IBM standard labels, the system rejects the tape and requests the operator to mount the tape volume specified.

THE PASSWORD AND NOPWREAD SUBPARAMETERS

The PASSWORD and NOPWREAD subparameters tell the system that you want the data set to be password protected. If you specify PASSWORD, the data set cannot be read from, written into, or deleted by another job step or job unless the operator can supply the system with the correct password. If you specify NOPWREAD (no password read), the data set can be read without the operator supplying the password, but the password is still required for writing or deleting data sets.

The PASSWORD and NOPWREAD subparameters are positional subparameters and must be coded third, after the data set sequence number subparameter and the label type subparameter or the commas that indicate their absence. If you want the data set password protected, specify PASSWORD when the data set is created. Password protected data sets must have standard labels, either IBM standard or American National Standard labels.

THE IN AND OUT SUBPARAMETERS

The basic sequential access method (BSAM) permits a specification of INOUT or OUTIN in the OPEN macro instruction as the processing method. If you have specified either of these processing methods in the OPEN macro instruction and want to override it, you may be able to do so by coding either the IN or OUT subparameter. For FORTRAN users, the IN and OUT subparameters provide a means of specifying how the data set is to be processed, i.e., for input or output.

When INOUT is specified in the OPEN macro instruction and you want the data set processed for input only, you can specify the IN subparameter. When the IN subparameter is coded, any attempt by the processing program to process the data set for output is treated as an error.

When OUTIN is specified in the OPEN macro instruction and you want the data set processed for output only, you can specify the OUT subparameter. When the OUT subparameter is coded, any attempt by the processing program to process the data set for input is treated as an error.

The IN and OUT subparameters are positional subparameters. If either is coded, it must appear as the fourth subparameter, after the data set sequence number subparameter, the label type subparameter, and the PASSWORD subparameter, or the commas that indicate their absence.

THE RETPD AND EXPDT SUBPARAMETERS

When it is necessary that a data set be kept for some period of time, you can tell the system how long it is to be kept when you create the data set. As long as the time period has not expired, a data set that resides on a direct access volume cannot be deleted by or overwritten by another job step or job. (If it is necessary to delete such a data set, you can use the IEHPROGM utility program to delete the data set. The IEHPROGM utility program is described in the Utilities publication.)

There are two different ways to specify a time period: (1) tell the system how many days you want the data set kept, the RETPD subparameter, or (2) tell the system the exact date after which the data set need no longer be kept, the EXPDT subparameter.

If you code the RETPD subparameter, you specify a 1- to 4-digit number, which represents the number of days the data set is to be kept. (Leap year is not considered when determining the retention period.) If you code the EXPDT subparameter, you specify a 2-digit year number and a 3-digit day number (e.g., January 1 would be 001, July 1 would be 182), which represents the date after which the data set need no longer be kept. When neither the RETPD or EXPDT subparameter is specified for a new data set, the system assumes a retention period of zero days.

To make sure that a temporary data set is deleted at the end of the job, you should not specify a retention period or expiration date, either directly on the LABEL parameter or indirectly by coding DCB=dsname to copy information from the label of a cataloged data set. If you do specify a retention period or expiration date for a temporary

data set, the system will not delete the data set until the time period has expired.

The RETPD or EXPDT subparameter must follow all other subparameters of the LABEL parameter. If no other subparameters are coded, you can code LABEL=RETPD=nnnn or LABEL=EXPDT=yyddd.

Examples of the LABEL Parameter

1.
```
//DD1 DD  DSNAME=HERBI,DISP=(NEW,KEEP),UNIT=TAPE,                    X
//       VOLUME=SER=T2,LABEL=(3,NSL,RETPD=188)
```

This DD statement defines a new data set. The LABEL parameter tells the system: (1) this data set is to be the third data set on the tape volume; (2) this data set has nonstandard labels; and (3) this data set is to be kept for 188 days.

2.
```
//DD2 DD  DSNAME=A.B.C,DISP=(,CATLG,DELETE),UNIT=2400-2,             X
//       LABEL=(,NL)
```

This DD statement defines a new data set and requests the system to catalog it. The catalog entry for this data set will not indicate that the data set has no labels. Therefore, each time this data set is referred to by a DD statement, the statement must include LABEL=(,NL).

3.
```
//DD3 DD  DSNAME=SPECS,UNIT=2400,VOLUME=SER=10222,                   X
//       DISP=OLD,LABEL=4
```

This DD statement defines an existing data set. The LABEL parameter indicates that the data set is the fourth data set on the tape volume.

4.
```
//STEP1   EXEC PGM=FIV
//DDX     DD   DSNAME=CLEAR,DISP=(OLD,PASS),UNIT=2400-4              X
//            VOLUME=SER=1257,LABEL=(,NSL)
//STEP2   EXEC PGM=BOS
//DDY     DD   DSNAME=*.STEP1.DDX,DISP=OLD,LABEL=(,NSL)
```

The DD statement named DDX in STEP1 defines an existing data set that has nonstandard labels and requests the system to pass the data set. The DD statement named DDY in STEP2 receives the passed data set. Unit and volume information is not specified since this information is available to the system; the label type is not available to the system and must be coded.

The OUTLIM Parameter

```
OUTLIM=number
```

number
the limit for the number of logical records you want included in the output data set being routed through the output stream. The maximum number that can be specified is 16777215.

Rules for Coding

1. The OUTLIM parameter has meaning only if the System Management facilities option with system, job, and step data collection was selected at system generation.

2. The OUTLIM parameter is ignored unless SYSOUT is coded in the operand field of the same DD statement.

3. The value specified for OUTLIM can be any number from 1 through 16777215.

4. If OUTLIM is not specified or if OUTLIM=0 is specified, no output limiting is done.

5. The OUTLIM and DDNAME parameters are mutually exclusive and should not be coded together.

What the OUTLIM Parameter Does

The OUTLIM parameter allows you to specify a limit for the number of logical records you want included in the output data set being routed through the output stream. When the number specified is reached, an exit provided by the System Management Facilities option is taken to a user supplied routine that determines whether to cancel the job or increase the limit. If the exit routine is not supplied, the job is cancelled.

Determining the Output Limit

The limit for the number of logical records you want as output must include a system overhead factor. Generally, the value you add to the limit is eight times the blocking factor for your data. (For those programmers who need a more precise value, the system overhead is the number of EXCPs issued each time the OPEN or CLOSE macro instruction is issued for the data set.)

References:

1. For information on coding the SYSOUT parameter on the DD statement, refer to the section "The SYSOUT PARAMETER -- MFT, MVT" in this publication.
2. A discussion of the System Management Facilities Option is contained in the Introduction publication. Information on user exit routines to be used with the System Management Facilities Option is contained in the publication System Management Facilities Guide.

Example of the OUTLIM Parameter

1. `//OUTPUT DD SYSOUT=F,OUTLIM=1000`

The limit for the number of logical records is 1000.

The QNAME Parameter -- MFT and MVT with TCAM

```
QNAME=process name
```

process name
specifies the name of a TPROCESS macro which defines a destination queue for messages that are to be processed by an application program and creates a process entry for the queue in the Terminal Table.

Rules for Coding

1. The process name must consist of 1 through 8 alphameric and national (#,$,@) characters. The first character must be an alphabetic or national character.

2. The process name is identical to the symbolic name on the TPROCESS macro.

3. The DCB parameter is the only parameter that can be coded on a DD statement with the QNAME parameter. BLKSIZE, BUFL, LRECL, OPTCD, and RECFM are the only operands that may be specified as subparameters. QNAME is mutually exclusive with DDNAME, DSNAME, and SYSOUT keywords. These subparameters are defined in the Glossary of DCB Subparameters in the section on the DCB parameter.

What the QNAME Parameter Does

The text portion of messages received from stations by means of the Telecommunications Access Method (TCAM) may be processed by an application program. The QNAME parameter is used to access these messages for the application program. Like the DUMMY parameter used with sequential access methods, the QNAME parameter does not perform input or output operations on a data set. The process name specified by the QNAME parameter names a TPROCESS macro which serves as the link between the Message Control Program (MCP) and an application program.

An application program exists as a separate system task or subtask in the same computer as the MCP. Messages that are to be processed are placed in a destination queue by the Message Handler of the MCP. The TPROCESS macro defines the destination queue and creates an entry for the queue (a process entry) in the Terminal Table. The user can indicate at execution time which destination queue is to be used by specifying a particular TPROCESS macro in the QNAME parameter on the DD statement.

Example of the QNAME Parameter

```
//DDY   DD   QNAME=FIRST,DCB=(RECFM=F,LRECL=80,BLKSIZE=320)
```

This DD statement is used in an application program to define data that is to be accessed by TCAM. "FIRST" is the name of the TPROCESS macro that specifies the destination queue through which messages that must be processed by the application program are routed. The DCB parameter is coded to supply information for the data control block that was not supplied in the DCB macro instruction.

The SEP Parameter

```
SEP=(ddname,...)
```

ddname
    the names of up to eight earlier DD statements in the same job step.

Rules for Coding

1. Each ddname must be separated by a comma.

2. If only one ddname is coded, you need not enclose it in parentheses.

3. If channel separation is critical, use the UNIT parameter to specify a particular channel, using an absolute address or group name. (How to specify a particular channel is described under "Unit Address" in the chapter "The UNIT Parameter.")

4. The SEP, AFF, DDNAME, and SYSOUT parameters are mutually exclusive parameters; therefore, when AFF, DDNAME, or SYSOUT is coded, do not code the SEP parameter.

The devices that the system allocates for data sets used in a job step are attached to channels. These channels transmit the data in the data sets from the device to the CPU. When two or more data sets are to be used in a job step, processing time may be shortened if the system transmits data over separate channels.

Requesting Channel Separation

The SEP and AFF parameters can be used to request channel separation. You list in the SEP parameter the names of up to eight earlier DD statements in the job step that define data sets from which channel separation is desired. Coding the AFF parameter is a shortcut method of requesting channel separation, since you refer to an earlier DD statement in the same job step that contains the SEP parameter. (The AFF parameter is described in the chapter "The AFF Parameter.")

If the system finds it impossible in the current environment to satisfy the channel separation request, the system may try to alter the current environment through some operator action. The operator is given the option of bringing a device online, cancelling the channel separation request, or cancelling the job. In certain environments, the operator may also be able to tell the system to wait for devices to become free. If you make a nonspecific request for a direct access volume and request channel separation, your request for separation may be ignored. This happens when the algorithm used to allocate data sets to devices is not able to select the device that would permit the desired channel separation.

Requests for channel separation are ignored for any data sets that have been allocated devices by the automatic volume recognition (AVR) option.

If it is essential that data be transmitted via a particular channel, you can specify an absolute unit address or group name (if the group of devices is associated with one channel) in the UNIT parameter.

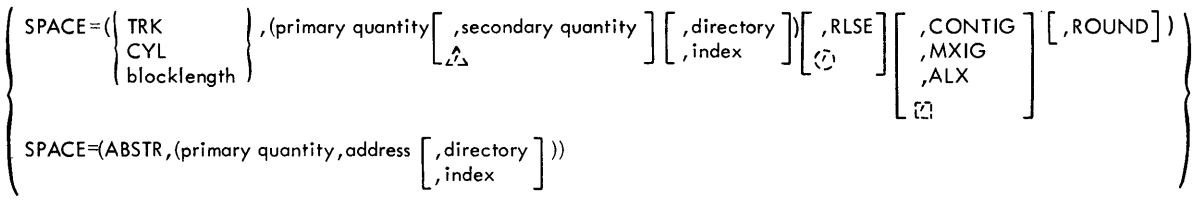
If neither the SEP nor AFF parameter is coded, any available channel, consistent with the UNIT parameter requirement, is assigned by the system.

Example of the SEP Parameter

```
1.  //STEP1   EXEC PGM=STARTS
    //DD1     DD   DSNAME=X.Y.Z,DISP=OLD
    //DD2     DD   DSNAME=&&WORK,DISP=(,PASS),UNIT=2311,                X
    //             SPACE=(CYL,(3,1))
    //DD3     DD   DSNAME=NABS,DISP=OLD,VOLUME=SER=7110,UNIT=2311
    //DD4     DD   DSNAME=PARE,DISP=OLD,VOLUME=SER=E59,                 X
    //             UNIT=2311,SEP=(DD2,DD3)
```

The system attempts to assign the data set defined by the DD statement named DD4 to a channel other than the ones assigned to the data sets defined by the DD statements DD2 and DD3. Since the SEP parameter did not include the ddname DD1, the data set defined by DD1 and the data set defined by DD4 may or may not be assigned to the same channel.

The SPACE Parameter



TRK
specifies that space is to be allocated by track.

CYL
specifies that space is to be allocated by cylinder.

DD

block length
specifies the average block length of the data. The system computes how many tracks to allocate.

primary quantity
specifies how many tracks or cylinders are to be allocated, or how many blocks of data are to be contained in the data set. If you also specify a number of tracks or cylinders to be used as a directory or index, the primary quantity will be the total number of tracks or cylinders assigned. (Example: if SPACE=(CYL,(10,1)) is specified, 10 cylinders will be assigned.)

,secondary quantity
specifies how many more tracks or cylinders are to be allocated if additional space is required, or how many more blocks of data may be included if additional space is required.

⚠
specifies that the system is not to allocate additional space if it is required, and either a directory space requirement or index space requirement follows.

,directory
specifies the number of 256-byte records that are to be contained in the directory of a partitioned data set.

,index
specifies the number of cylinders that are required for the index of an indexed sequential data set.

,RLSE
specifies that space allocated to the data set that is not used is to be released.

(0)
specifies that space allocated to the data set that is not used is not to be released and another subparameter follows.

,CONTIG
specifies that space allocated to the data set must be contiguous.

,MXIG
specifies that the space allocated to the data set must be the largest area of contiguous space on the volume and the space must be equal to or greater than the space requested. This subparameter applies only to the primary space allocation.

,ALX
specifies that up to five different contiguous areas of space are to be allocated to the data set and each area must be equal to or greater than the space requested.

specifies that CONTIG, MXIG, or ALX is not specified and the ROUND subparameter follows.

,ROUND
specifies that space is requested by specfying the average block length of the data and the space allocated to the data set must be equal to one or more cylinders.

ABSTR
specifies that the data set is to be placed at a specific location on the volume.

primary quantity
specifies the number of tracks to be allocated to the data set.

address
specifies the track number of the first track to be allocated.

,directory
specifies the number of 256-byte records that are to be contained in the directory of a partitioned data set.

,index
specifies the number of tracks that are required for the index of an indexed sequential data set. The number of tracks must be equal to one or more cylinders.

Rules for Coding

1. The SPACE parameter has no meaning for tape volumes; however, if a data set is assigned to a device class that contains both direct access devices and tape devices, e.g., UNIT=SYSSQ, the SPACE parameter should be coded.

2. If you do not code secondary, directory, or index quantities, you need not enclose the primary quantity in parentheses.

3. Code the second format of the SPACE parameter when you want a data set placed in a specific position on a direct access device.

4. The SPACE, SPLIT, SUBALLOC, and DDNAME parameters are mutually exclusive parameters; therefore, if SPLIT, SUBALLOC, or DDNAME is coded, do not code the SPACE parameter.

REQUESTING SPACE FOR A DATA SET

Every data set that is to be written on a direct access volume must be allocated space on the volume before the data set can be written. There are three different parameters that can be used to request space -- SPACE, SPLIT, SUBALLOC -- and they are mutually exclusive. The SPLIT and SUBALLOC parameters are discussed in the chapters "The SPLIT Parameter" and "The SUBALLOC Parameter," respectively.

SPECIFYING THE SPACE PARAMETER

Space for data sets is allocated before the job step is executed. If a request for space cannot be satisfied, the job is terminated.

There are two different ways to code the SPACE parameter. One way tells the system how much space you want and lets the system assign specific tracks. The other way tells the system the specific tracks you want.

Letting the System Assign Specific Tracks

When you want the system to assign specific tracks, you must specify in the SPACE parameter:

- The unit of measurement the system should use for allocating space; specify TRK, for tracks, CYL, for cylinders, or the average block length of the data, for blocks.
- The amount of space to be allocated; specify the primary quantity as a number of tracks, cylinder, or blocks.

Optionally, you can specify in the SPACE parameter:

- That additional space is to be allocated to the data set if it is required; specify a secondary quantity of tracks, cylinders, or blocks.
- The size of a directory or index area; specify the number of records required for a directory or the number of cylinders required for an index.
- That unused space is to be released; specify the RLSE subparameter.
- The format of the space allocated to the data set; specify the CONTIG, MXIG, or ALX subparameter.
- That space is to begin with a cylinder; specify the ROUND subparameter.

When a disk operating system (DOS) volume is mounted for use in an IBM System/360 Operating System, you can let the system assign specific tracks on the DOS volume for a new data set. (There are restrictions on the use of an existing DOS data set in an IBM System/360 Operating System; these restrictions are described in the Data Management for System Programmers volume.

SPECIFYING THE UNIT OF MEASUREMENT

The first subparameter of the SPACE parameter identifies the unit of measurement to be used in allocating the data set and can be specified as:

- TRK -- if you want space allocated by track.
- CYL -- if you want space allocated by cylinder. CYL must be specified if you are creating an indexed sequential data set.
- a number of bytes which represents the average block length of the data -- if you want the system to compute and allocate the least number of tracks required to contain the blocks.

Since the next subparameter (primary quantity) tells the system how many of these units you require, specify the unit that makes it most convenient for you to express your space requirement. A request for cylinders (CYL) provides the most efficient performance.

When you request space in units of blocks, the average block length cannot exceed 65,535. If the blocks have keys, code the DCB subparameter KEYLEN on the DD statement and specify the key length, i.e., DCB=KEYLEN=key length.

SPECIFYING A PRIMARY QUANTITY

The primary quantity tells the system how many tracks or cylinders are to be allocated to the data set or how many blocks of data will be written. (Note: You must consider track overflow when computing track requirements.) When the first subparameter of the SPACE parameter specifies the average block length, the system computes the number of tracks (or cylinders if the ROUND subparameter is coded) required based on the number of blocks specified as the primary quantity.

There must be enough available space on one volume to satisfy the primary quantity. If you make a specific request for a single volume and there is not enough space on that volume to satisfy the space request, the job step is abnormally terminated. If you specify multiple volume serial numbers, the system will search each volume until it finds a volume with sufficient space or until it determines that none of the volumes specified contain enough space. If none of the volumes contains enough space to satisfy the request, the job step will be terminated. If you make a nonspecific volume request, i.e., no volume serial numbers are specified on the DD statement, the system selects a mounted volume or causes a volume to be mounted and then determines if there is enough space available on the volume to satisfy the request for space. If there is not enough space available, the system selects another volume. Allocation may wait for space to become available on specific and nonspecific requests if other tasks have data sets allocated on the eligible volumes. Care should be taken in requesting a primary quantity, since an invalid request (e.g., a quantity greater than physical device limits) will cause the system to attempt to find space on all eligible units. This can cause unnecessary mounting of volumes or a wait for space that cannot be satisfied. In the latter case it is necessary to cancel the job.

The system attempts to allocate the primary quantity in contiguous tracks or cylinders. If contiguous space is not available, the system satisfies the space request with up to five noncontiguous blocks (extents) of space. If a user label is requested, the system allocates up to four noncontiguous blocks of space. You can override these system actions by coding the CONTIG, MXIG, or ALX subparameter; these subparameters are discussed later.

If the system assigns a temporary data set to a dedicated data set, the primary quantity specified for the temporary data set is ignored. The system allocates the primary quantity requested on the DD statement defining the dedicated data set. A secondary quantity specified for the temporary data set will, however, override any secondary quantity specified on the DD statement defining the dedicated data set. (For detailed information on dedicated data sets, refer to the chapter "System Reader, Initiator and Writer Catalogs Procedures" in the publication Data Management for System Programmers.)

SECONDARY QUANTITY

The secondary quantity (incremental quantity) tells the system that you want additional space allocated to the data set if it is required. You specify as the secondary quantity how many more tracks or cylinders you want allocated or how many more blocks of data may be written. (When you request space in units of blocks, the system computes the number of tracks required for the primary quantity based on the average block length that you specified in the SPACE parameter. The system computes the number of tracks required for the secondary quantity based on what is specified in the DCB subparameter BLKSIZE. Therefore, include the DCB subparameter BLKSIZE on the DD statement, i.e., DCB=BLKSIZE=maximum block length.) Specifying a secondary quantity is optional.

If you do specify a secondary quantity and the data set requires additional space, the system allocates this space based on the quantity you specified. The system attempts to allocate the secondary quantity in contiguous tracks or cylinders. If contiguous space is not available, the system attempts to allocate the secondary quantity in up to five noncontiguous blocks (extents) of space.

Each time the data set requires more space, the system allocates the secondary quantity. This space is allocated on the same volume on which the primary quantity was allocated until: (1) there is not enough space available on the volume to allocate the secondary quantity, or (2) a total of 16 extents have been allocated to the data set. If either of these conditions is satisfied, the system must allocate the secondary quantity on another volume. You can specify this in one of two ways:

- For a specific volume request, specify more than one volume in the VOLUME parameter and request more volumes than devices.
- For a non-specific volume request, code PRIVATE and specify more than one volume in the VOLUME parameter.

If there is no more space available on the volumes that you requested, and if at least one volume is demountable, the system will request scratch volumes to be mounted until either the data set is complete or until all entries in the JFCB are filled. If the entries in the JFCB are already filled or if there is no demountable volume, the job step will abnormally terminate.

If a data set has used all the primary space allocated to it, a later job step can lengthen the data set with additional output by requesting a secondary quantity. You can specify a secondary quantity for an old data set whether or not you specified a secondary quantity when the data set was created. If you did specify a secondary quantity when you created the data set, you can override that quantity by specifying a different secondary quantity when you extend the data set. The secondary quantity you specify when you extend the data set is in effect only for the duration of the job step.

For indexed sequential data sets, a secondary quantity cannot be requested. If you request a secondary quantity for a checkpoint data set, the space cannot be used for a successful completion of the checkpoint entry. To determine how the space is used, refer to the chapter "Checkpoint and Restart" in the Supervisor and Macro Services Instructions publication.

The secondary quantity is a positional subparameter. If you specify a secondary quantity, the quantity must follow the primary quantity. If you do not specify a secondary quantity and specify the size of an index or directory as the next subparameter, you must code a comma to indicate the absence of a secondary quantity.

REQUESTING SPACE FOR A DIRECTORY OR INDEX

If the data set you are creating is a partitioned data set, you must request the system to allocate space for a directory. A directory consists of 256-byte records, and you specify, in the SPACE parameter, how many of these records the directory is to contain. These records contain entries for the members of the partitioned data set. You can determine how many records you should request for the directory by referring to the chapter "Processing a Partitioned Data Set" in the Data Management Services publication.

If the data set you are creating is an indexed sequential data set, you can tell the system, in the SPACE parameter, how many cylinders to allocate for the index. (The alternate way to request space for the index is to include, as one of the DD statements used to define an indexed sequential data set, a DD statement that defines the index and specifies the number of cylinders required for the index as the primary quantity.)

The system can differentiate between a specification of the number of records for a directory and the number of cylinders for an index by examining the DCB parameter on the DD statement. Any DD statement that defines an indexed sequential data set must include the DCB subparameter DSORG=IS or DSORG=ISU. When neither is specified, the system assumes you are requesting space for a directory.

RELEASING UNUSED SPACE -- RLSE

The RLSE subparameter allows you to request the system to delete unused space when the data set is closed. The space is released according to the units of space specified on the current DD statement for the data set. If you requested space in units of tracks, any unused tracks are released. If you requested space in units of cylinders, any unused cylinders are released. If you requested space in units of blocks, any unused tracks or cylinders (if ROUND was specified) are released. Note that the current DD statement's units of space can differ from the units specified during the initial allocation of the data set.

If you code the SPACE parameter on the DD statement that defines an output data set and includes the RLSE subparameter, the data set's unused space is released. Closing an output data set will not release space unless RLSE is specified on the current DD statement for the data set even if RLSE was specified when the data set was initially allocated.

If you code the SPACE parameter on a DD statement that defines an existing data set and include the RLSE subparameter, the data set's unused space is released.

If you have specified RLSE and the job step abnormally terminates, unused space is not released.

The RLSE subparameter is a positional subparameter. If you omit the RLSE subparameter and another subparameter follows, indicate the absence of the RLSE subparameter with a comma.

The RLSE subparameter is ignored when the TYPE=T option is coded in the CLOSE macro instruction.

SPECIFYING THE FORMAT OF ALLOCATED SPACE -- CONTIG, MXIG, OR ALX

The system attempts to allocate space in contiguous tracks or cylinders. If contiguous space is not available, the system satisfies the space request with up to five noncontiguous blocks of space. If a user label is requested, the system allocates up to four noncontiguous blocks of space. You can override these system actions by coding the CONTIG, MXIG, or ALX subparameter.

The CONTIG (contiguous) subparameter tells the system that the space it allocates to a data set must be contiguous. If the request cannot be satisfied, the job is terminated. If secondary space is allocated to the data set, it may not be contiguous to the original space allocated to the data set.

The MXIG (maximum contiguous) subparameter tells the system to allocate the largest area of contiguous space available on the volume. The area must be at least as large as the primary quantity requested. The MXIG subparameter cannot be specified for an indexed sequential data set.

The ALX (all extents) subparameter tells the system to allocate up to five different areas of contiguous space. If a user label is requested, the system allocates up to four different areas of contiguous space. Each area is to be at least as large as the primary quantity you requested. The system allocates as many areas, up to the maximum, as are available. The ALX subparameter cannot be specified for an indexed sequential data set.

Whichever of these subparameters you choose must follow either the RLSE subparameter or the comma that indicates its absence. If you do not specify one of these subparameters and the ROUND subparameter follows, indicate the absence of the CONTIG, MXIG, and ALX subparameters with a comma.

ALLOCATING WHOLE CYLINDERS -- ROUND

When you request space in units of blocks, you can request that the allocated space be equal to one or more cylinders. To request this, code ROUND as the last subparameter in the SPACE parameter. The system computes the number of tracks required to hold the blocks, and ensures that the space begins on the first track of a cylinder and ends on the last track of a cylinder.

Assigning Specific Tracks

You can place a data set in a specific position on a direct access volume by specifying in the SPACE parameter:

- ABSTR as the first subparameter.
- How many tracks you want allocated.
- The relative track number of the beginning track on which you want the data set placed.

If the data set is a partitioned data set, you must also specify how many records you want allocated for a directory. If the data set is an indexed sequential data set, you can also indicate how many tracks are required for the index. (The number of tracks you specify must be equal to one or more cylinders, and any other DD statement used to define the indexed sequential data set must specify ABSTR in the SPACE parameter. If either of these conditions is not met, the job is terminated.)

To determine the relative track number, count the first track of the first cylinder on the volume as 0, and count through the tracks on each cylinder until you reach the track on which you want your data set to start. (Track 0 cannot be requested.) The system automatically converts the relative track number to an address; this address varies with different devices. For indexed sequential data sets, the relative track number must correspond to the first track on a cylinder. Capacities of a number of direct access devices are listed in "Data Set Disposition and Space Allocation" in the Data Management Services publication.

If the tracks you request have already been allocated to another data set, the job is terminated.

Examples of the SPACE Parameter

1.
```
//DD1 DD  DSNAME=&&TEMP,UNIT=MIXED,SPACE=(CYL,10)
```

This DD statement defines a temporary data set and requests the system to assign any available tape or direct access volume (UNIT=MIXED specifies a group name of units that consists of tape and direct access devices). If a tape volume is assigned, the SPACE parameter is ignored; if a direct access volume is assigned, the SPACE parameter is used to allocate space to the data set. The SPACE parameter includes only the required subparameters (i.e., the type of units and a primary quantity), and requests the system to allocate 10 cylinders.

2.
```
//DD2 DD  DSNAME=ELLN,DISP=(,KEEP),UNIT=2314,                    X
//        VOLUME=SER=11257,SPACE=(1024,(100,25),,,ROUND),        X
//        DCB=BLKSIZE=2048
```

This DD statement defines a new data set that is to be written on a direct access volume. The SPACE parameter requests the system to compute the space required for the primary quantity; the system computes the space required based on an average block length of 1024 bytes, and up to 100 blocks of data will be written. If more space is required, the system is to compute how much additional space to allocate; the system computes the space required based on a maximum block length of 2048 bytes (specified in the BLKSIZE subparameter), and up to 25 blocks of data will be written. Since the ROUND subparameter is coded, the system ensures that the allocated space begins on the first track of a cylinder and ends on the last track of a cylinder.

3.
```
//DD3 DD  DSNAME=PDS12,DISP=(,KEEP),UNIT=2311,                   X
//        VOLUME=SER=26143,SPACE=(TRK,(200,,10),,CONTIG)
```

This DD statement defines a new partitioned data set. The system allocates tracks to the data set and 10 256-byte records for a directory. Since the CONTIG subparameter is coded, the system allocates 200 contiguous tracks on the volume.

4.
```
//DD4 DD  DSNAME=INDSEQ(INDEX),UNIT=2314,DCB=DSORG=IS,           X
//        DISP=(,KEEP),SPACE=(ABSTR,(20,40))
```

This DD statement defines the index area for a new indexed sequential data set. The SPACE parameter allocates 20 tracks (for a 2314, 20 tracks equal 1 cylinder), beginning with the fortieth track on the volume (the fortieth track on the volume is the beginning of the third cylinder).

Device	Storage Medium	Cylinders	Tracks Per Cylinder	Bytes Per		
				Track	Cylinder	Device (in millions)
2301	Drum	25*	8	20,483	4.09 (million)	4.09
2302	Disk	Model 3: 492 Model 4: 984	46	4,984	229,264	Model 3: 112.79 Model 4: 225.59
2303	Drum	80	10	4,892	48,920	3.9
2311	Disk	200	10	3,625	36,250	7.25
2314/2319 (each volume)	Disk	200	20	7,294	145,880	29.17
2321	Strip of Tape	980**	20	2,000	40,000	39.2

*There are 25 logical cylinders in a 2301 Drum.
**A volume is equal to one bin in a 2321 Data Cell.

Figure 23. Direct Access Capacities

Maximum Bytes per Record Formatted without Keys									Records per Track	Maximum Bytes per Record Formatted with Keys							
2311	2314/2319	2302	2303	2301	2321	2305-1	2305-2	3330		2311	2314/2319	2302	2303	2301	2321	2305-1	2305-2	3330
3625	7294	4984	4892	20483	2000	14136	14660	13030	1	3605	7249	4964	4854	20430	1984	13934	14569	12974
1740	3520	2403	2392	10175	935	6852	7231	6447	2	1720	3476	2383	2354	10122	920	6650	7140	6391
1131	2298	1570	1558	6739	592	4424	4754	4253	3	1111	2254	1550	1520	6686	576	4222	4663	4197
830	1693	1158	1142	5021	422	3210	3516	3156	4	811	1649	1139	1104	4968	406	3008	3425	3100
651	1332	912	892	3990	320	2480	2773	2498	5	632	1288	893	854	3937	305	2278	2682	2442
532	1092	749	725	3303	253	1996	2278	2059	6	512	1049	730	687	3250	238	1794	2187	2003
447	921	634	606	2812	205	1648	1924	1745	7	428	877	614	568	2759	190	1446	1833	1689
384	793	546	517	2444	169	1388	1659	1510	8	364	750	527	479	2391	154	1186	1568	1454
334	694	479	447	2157	142	1186	1452	1327	9	315	650	460	409	2104	126	984	1361	1271
295	615	425	392	1928	119	1024	1287	1181	10	275	571	406	354	1875	103	822	1196	1125
263	550	381	346	1741	101	892	1152	1061	11	244	506	362	308	1688	85	690	1061	1005
236	496	344	308	1585	86	782	1040	962	12	217	452	325	270	1532	70	580	949	906
213	450	313	276	1452	73	688	944	877	13	194	407	294	238	1399	58	486	853	821
193	411	286	249	1339	62	608	863	805	14	174	368	267	211	1286	47	406	772	749
177	377	264	225	1241	53	538	792	742	15	158	333	245	187	1188	38	336	701	686
162	347	244	204	1155	44	478	730	687	16	143	304	224	166	1102	29	276	639	631
149	321	225	186	1079	37	424	676	639	17	130	277	206	148	1026	21	222	585	583
138	298	209	169	1012	30	376	627	596	18	119	254	190	131	959	15	174	536	540
127	276	196	155	952	24	334	584	557	19	108	233	176	117	899	9	132	493	501
118	258	183	142	897	20	296	544	523	20	99	215	163	104	844		94	453	467
109	241	171	130	848	15	260	509	491	21	90	198	152	92	795		58	418	435
102	226	161	119	804	10	230	477	463	22	82	183	142	81	751			386	407
95	211	151	109	763	6	200	448	437	23	76	168	132	71	710			357	381
88	199	143	100	726		174	421	413	24	69	156	123	62	673			330	357
82	187	135	92	691		150	396	391	25	63	144	116	54	638			305	335
77	176	127	84	659		128	373	371	26	58	133	108	46	606			282	315
72	166	121	77	630		106	352	352	27	53	123	102	39	577			261	296
67	157	114	70	603		88	332	335	28	48	114	95	32	550			241	279
63	148	108	64	577		70	314	318	29	44	105	89	26	524			223	262
59	139	102	58	554		52	297	303	30	40	96	83	20	501			206	247

Figure 24. Track Capacities

The SPLIT Parameter

```
SPLIT=  { (n,CYL,(primary quantity[,secondary quantity]))                    }
        { n                                                                  }
        { (percent,block length,(primary quantity[,secondary quantity]))     }
        { percent                                                            }
```

n
    specifies the number of tracks per cylinder you want allocated to the first data set.

CYL
    specifies that space is to be allocated by cylinder.

primary quantity
    specifies how many cylinders are to be allocated for use by all the associated data sets.

,secondary quantity
    specifies how many more cylinders are to be allocated to a data set if additional space is required.

n
    the number of tracks per cylinder you want allocated to the data set defined on the DD statement.

percent
    the percentage of tracks per cylinder you want allocated to the first data set, a number from 1 through 99.

block length
    specifies the average block length of the data. The system computes how many cylinders to allocate.

primary quantity
    specifies the total number of blocks to be allocated for use by all the associated data sets.

,secondary quantity
    specifies how many more blocks are to be allocated to a data set if additional space is required.

percent
    the percentage of tracks per cylinder you want allocated to the data set defined on the DD statement.

Rules for Coding

1. The first DD statement that contains the SPLIT parameter must contain volume and unit information. You need not code volume and unit information on the following DD statements that contain the SPLIT parameter.

2. If a secondary quantity is not specified, you need not enclose the primary quantity in parentheses.

3. The SPLIT, SPACE, SUBALLOC, DDNAME, and SYSOUT parameters are mutually exclusive parameters; therefore, if SPACE, SUBALLOC, DDNAME, or SYSOUT is coded, do not code the SPLIT parameter.

REQUESTING SPACE FOR A DATA SET

Every data set that is to be written on a direct access volume must be allocated space on the volume before the data set can be written. There are three different parameters that can be used to request space -- SPLIT, SPACE, SUBALLOC -- and they are mutually exclusive. The SPACE and SUBALLOC parameters are discussed in the chapters "The SPACE Parameter" and "The SUBALLOC Parameter," respectively.

Specifying the SPLIT Parameter

The SPLIT parameter is specified when data sets defined in a job step require space on the same volume, and you want to minimize access-arm movement by having the data sets share cylinders. The device on which the volume is mounted is said to be operating in a split cylinder mode when the SPLIT parameter is specified. In this mode, two or more data sets are stored so that portions of each data set occupy tracks within every allocated cylinder.

The cylinders allocated to the data sets must be on one volume. If there are not enough cylinders available on the volume to satisfy the request, the job is terminated. The SPLIT parameter cannot be used to allocate space for direct, partitioned, and indexed sequential data sets. If the SPLIT parameter is used to allocate space for data sets that are to reside on a drum storage volume, space is allocated for the data sets, but the data sets are not stored using the split cylinder mode. The space occupied by a data set residing on a cylinder that has been split is not available for reallocation until all data sets sharing the cylinder have been deleted.

The data sets that are to share cylinders are defined by a sequence of DD statements. The first DD statement in the sequence specifies the total amount of space required for all the data sets and the portion of that space required by this data set. Each succeeding DD statement in the sequence requests a portion of the total space.

In the SPLIT parameter, there are two ways to request the total amount of space for data sets that are to share cylinders. You can request the space in units of cylinders or in units of blocks.

REQUESTING SPACE IN UNITS OF CYLINDERS

When you request space in units of cylinders, the first DD statement of the sequence specifies in the SPLIT parameter:

- The number of tracks per cylinder to be allocated to this data set; specify a number.
- Space is to be allocated in units of cylinders; specify CYL.
- How many cylinders are to be allocated for use by all the data sets; specify the primary quantity as a number of cylinders.

Optionally, you can specify:

- That additional cylinders are to be allocated to a data set if additional space is required; specify the secondary quantity as a number of cylinders.

Each succeeding DD statement in the sequence specifies a number of tracks per cylinder or number of tracks per cylinder depending upon the preceding DD statement in the sequence, to be allocated to the data set.

If a secondary quantity (incremental quantity) is specified in the SPLIT parameter on the first DD statement in the sequence, any data set that exceeds its allocated space is allocated additional space in the amount of the secondary quantity. This additional space is allocated only to the data set that requires it and the space is not split with the other data sets. If a secondary quantity is not specified and a data set exceeds its allocated space, the job step is terminated.

REQUESTING SPACE IN UNITS OF BLOCKS

When you request space in units of blocks, the first DD statement of the sequence specifies in the SPLIT parameter:

- The percentage of tracks per cylinder to be allocated to this data set; specify a number from 1 to 99.
- The average block length of the data in the data sets; specify the average block length in bytes.
- How many blocks are to be allocated for use by all the data sets; specify the primary quantity as a number of blocks.

Optionally, you can specify:

- That additional blocks are to be allocated to a data set if additional space is required; specify the secondary quantity as a number of blocks.

Each succeeding DD statement in the sequence specifies a percentage of tracks per cylinder or number of tracks per cylinder depending upon the preceding DD statement in the sequence, to be allocated to the data set.

When a percentage of tracks per cylinder is requested, the system rounds down to the next full track when calculating the actual number of tracks to be allocated. If the percentage is less than 1 track, it will cause a JCL error.

When you request space in units of blocks, the system computes for you how many cylinders are required. The average block length cannot exceed 65,535 bytes. If the blocks have keys, code the DCB subparameter KEYLEN on the DD statement and specify the key length, i.e., DCB=KEYLEN=key length.

If a secondary quantity (incremental quantity) is specified in the SPLIT parameter on the first DD statement in the sequence, any data set that exceeds its allocated space is allocated additional space. The secondary quantity is specified as a number of blocks, and the system computes how many cylinders to allocate based on this number. This additional space is allocated only to the data set that requires it and the space is not split with the other data sets. If a secondary quantity is not specified and a data set exceeds its allocated space, the job step is terminated.

Examples of the SPLIT Parameter

1.
```
//STEP1   EXEC PGM=CREATE
//DD1     DD   DSNAME=QUEST,DISP=(,KEEP),UNIT=2311,                X
//            VOLUME=SER=757500,SPLIT=(3,CYL,(30,1))
//DD2     DD   DSNAME=APP,DISP=(,KEEP),SPLIT=4
//DD3     DD   DSNAME=SET,DISP=(,KEEP),SPLIT=3
```

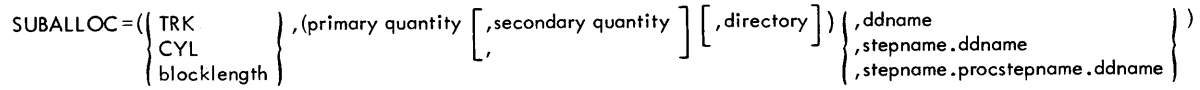
This job step contains a sequence of DD statements that define new data sets and request that these data sets share the same cylinders.

The first DD statement of the sequence, named DD1, specifies: (1) three tracks per cylinder are to be allocated to this data set; (2) space is to be allocated in units of cylinders; (3) thirty cylinders are to be allocated for use by all the data sets; and (4) any data set that exceeds the space allocated to it should be allocated another cylinder. The DD statement named DD2 requests that the system allocate 4 tracks per cylinder to this data set. The DD statement named DD3 requests that the system allocate 3 tracks per cylinder to this data set.

2.
```
//STEP2   EXEC PGM=PAGE
//DDX     DD   DSNAME=ISSA,DISP=(,KEEP),UNIT=2314,                X
//            VOLUME=SER=49463,SPLIT=(18,1024,(700))
//DDY     DD   DSNAME=SEL12,DISP=(,KEEP),SPLIT=48
//DDZ     DD   DSNAME=SEVE,DISP=(,KEEP),SPLIT=34
```

This job step contains a sequence of DD statements that define new data sets and request that these data sets share the same cylinders. The first DD statement of the sequence, named DDX, specifies in the SPLIT parameter: (1) 18 per cent of the tracks per cylinder are to be allocated to this data set; (2) the system is to compute how many cylinders are to be allocated for use by all the data sets based on an average block length of 1024 bytes and 700 blocks are required. The DD statement named DDY requests that the system allocate 48 per cent of the tracks per cylinder to this data set. The DD statement named DDZ requests that the system allocate 34 per cent of the track per cylinder to this data set. Since the first DD statement in the sequence does not specify a secondary quantity, the job is abnormally terminated when any of the data sets exceeds its allocated space.

The SUBALLOC Parameter



TRK
speciifes that space is to be allocated by track.

CYL
specifies that space is to be allocated by cylinder.

block length
specifies the average block length of the data. The system computes how many tracks to allocate.

primary quantity
specifies how many tracks or cylinders are to be allocated, or how many blocks of data are to be contained in the data set.

,secondary quantity
specifies how many more tracks or cylinders are to be allocated if the additional space is required, or how many more blocks of data may be included if additional space is required.

,
specifies that the system is not to allocate additional space if it is required, and a directory space requirement follows.

,directory
specifies the number of 256-byte records that are to be contained in the directory of a partitioned data set.

,ddname
specifies that the system must allocate space from the data set defined on the earlier DD statement named "ddname" that appears in the same job step.

,stepname.ddname
specifies that the system must allocate space from the data set defined on the DD statement named "ddname", which is contained in an earlier job step named "stepname" that is part of the same job.

,stepname.procstepname.ddname
specifies that the system must allocate space from the data set defined on the DD statement "ddname," which is contained in an earlier procedure step named "procstepname"; the procedure step is part of a cataloged or in-stream procedure called by an earlier job step named "stepname" that is part of the same job.

Rules for Coding

1. Before you can use the SUBALLOC parameter, you must define a new data set and request enough contiguous space in the SPACE parameter to contain all of the data sets.

2. When you code the SUBALLOC parameter, omit the VOLUME and UNIT parameters.

3. The SUBALLOC, SPACE, SPLIT, DDNAME, and SYSOUT parameters are mutually exclusive parameters; therefore, when SPACE, SPLIT, DDNAME, or SYSOUT is coded, do not code the SUBALLOC parameter.

REQUESTING SPACE FOR A DATA SET

Every data set that is to be written on a direct access volume must be allocated space on the volume before the data set can be written. There are three different parameters that can be used to request space -- SUBALLOC, SPACE, SPLIT -- and they are mutually exclusive. The SPACE and SPLIT parameters are discussed in the chapters "The SPACE Parameter" and "The SPLIT Parameter," respectively.

Specifying the SUBALLOC Parameter

The SUBALLOC parameter allows you to place a series of data sets on one volume and in a certain sequence, in a contiguous area of space. This area of space is first allocated to one data set, then later DD statements defining new data sets in the same job may request parts of this space. This is called suballocation. Suballocation is used to minimize access-arm movement when data sets are processed serially. The SUBALLOC parameter cannot be used to allocate space for an indexed sequential data set.

To use suballocation, you must first define a data set on a DD statement and use the SPACE parameter to request space. This data set must be used only for suballocation purposes, i.e., the data set should contain no data. The space you request must be large enough to contain all of the data sets and the space must be contiguous. On this same DD statement, you can request more than one device in the UNIT parameter or more than one volume in the VOLUME parameter. This allows a suballocated data set for which a secondary quantity was requested in the SUBALLOC parameter to be continued on another volume if the data set exceeds its primary quantity.

Once this data set has been defined, other data sets defined in the job can use the previously allocated space by specifying the SUBALLOC parameter. Each DD statement that specifies the SUBALLOC parameter causes the new data set to be assigned to the next area of unused space from the original data set.

You must specify in the SUBALLOC parameter:

- The unit of measurement the system should use for allocating space; specify TRK, for tracks, CYL, for cylinders, or the average block length of the data, for blocks.
- The amount of space to be allocated; specify the primary quantity as a number of cylinders, tracks, or blocks.
- Where in the job the original data set is defined; specify the name of the DD statement that defines the data set and the name of the job step in which the DD statement appears.

Optionally, you can specify in the SUBALLOC parameter:

- That additional space is to be allocated to the data set if it is required; specify a secondary quantity of tracks, cylinders, or blocks.
- The size of a directory; specify the number of records required for a directory.

SPECIFYING THE UNIT OF MEASUREMENT

The first subparameter of the SUBALLOC parameter identifies to the system the unit of measurement to be used in suballocating space for the data set and can be specified as:

- TRK -- if you want space suballocated by track.
- CYL -- if you want space suballocated by cylinder.
- a number of bytes, which represents the average block length of the data -- if you want the system to compute and allocate the least number of tracks required to contain the blocks.

Since the next subparameter tells the system how many of these units you require, specify the unit that makes it most convenient for you to express your space requirement. A request for cylinders (CYL) provides the most efficient performance. Since suballocation works with contiguous tracks or cylinders, when a data set suballocates space from another data set, the space suballocated starts at the first track available. If this track is not on a cylinder boundary, the data set will be allocated in tracks equivalent to the number of cylinders specified.

When you request space in units of blocks, the average block length cannot exceed 65,535 bytes. If the blocks have keys, you must specify the key length in the DCB subparameter KEYLEN=n.

SPECIFYING A PRIMARY QUANTITY

The primary quantity tells the system how many tracks or cylinders are to be suballocated for the data set or how many blocks of data will be written. If there is not enough space available in the original data set to satisfy the primary quantity request, the job is terminated. When the first subparameter of the SUBALLOC parameter specifies the average block length, the system computes the number of tracks required based on the number of blocks specified as the primary quantity.

IDENTIFYING THE ORIGINAL DATA SET

In the SUBALLOC parameter, you must identify the master data set from which space is to be suballocated. Space can be suballocated from an existing master data set -- i.e., the master data set need not be created in the same job as you create a data set to be suballocated. However, the job must include a DD statement defining the master data set. Refer to this DD statement in the SUBALLOC parameter by coding:

- ddname -- if the DD statement defining the master data set appears in the same job step.
- stepname.ddname -- if the DD statement appears in an earlier job step.
- stepname.procstepname.ddname -- if the DD statement appears in a procedure step that is part of a cataloged or in-stream procedure called by an earlier job step.

SPECIFYING A SECONDARY QUANTITY

The secondary quantity (incremental quantity) tells the system that you want additional space allocated to the data set if it is required. You specify as the secondary quantity how many more tracks or cylinders you want allocated or how many more blocks of data may be written. (When you

request space in units of blocks, the system computes the number of tracks required for the primary quantity based on the average block length that you specified in the SPACE parameter. The system computes the number of tracks required for the secondary quantity based on what is specified in the DCB subparameter BLKSIZE. Therefore, include the DCB subparameter BLKSIZE on the DD statement, i.e., DCB=BLKSIZE=maximum block length.) Specifying a secondary quantity is optional.

If you specify a secondary quantity and the data set requires additional space, the system allocates this space based on the quantity you specified. This additional space is allocated from available space on the volume, not from the space in the original data set from which the system suballocated space for this data set. If more than one device or volume was requested on the same DD statement that requested space for suballocation, the data set can be continued onto another volume.

A data set may use all the space allocated to it and a later job step or job may then try to lengthen the data set with additional output. In this case, the data set can be lengthened only if a secondary quantity was specified when the data set was created and only if there is enough space available on the volume. If a later job step or job is lengthening a data set and specifies a secondary quantity, this quantity overrides, for the duration of the step, any secondary quantity specified when the data set was created.

The secondary quantity is a positional subparameter. If you specify a secondary quantity, the quantity must follow the primary quantity. If you do not specify a secondary quantity and specify the size of a directory as the next subparameter, you must code a comma to indicate the absence of a secondary quantity.

REQUESTING SPACE FOR A DIRECTORY

If the data set you are creating is a partitioned data set, you must request that the system allocate space for a directory. A directory consists of 256-byte records and you specify how many of these records the directory is to contain. These records contain entries for the members of the partitioned data set. You can determine how many records you should request for the directory by referring to the chapter "Processing a Partitioned Data Set" in the Data Management Services publication.

If you request space for a directory in the SUBALLOC parameter, the request must follow the secondary quantity or the comma that indicates its absence.

Examples of the SUBALLOC Parameter

```
1.  //STEP1   EXEC PGM=PREP
    //DD1     DD   DSNAME=DUM,DISP=(,KEEP),UNIT=2302,                  X
    //           VOLUME=SER=ALLDS,SPACE=(CYL,50,,CONTIG)
    //STEP2   EXEC PGM=BSPED
    //DD2     DD   DSNAME=SPEC50,DISP=(,KEEP),                        X
    //           SUBALLOC=(CYL,(20,1),STEP1.DD1)
    //DD3     DD   DSNAME=SPEC51,DISP=(,KEEP),                        X
    //           SUBALLOC=(TRK,(44,7),STEP1.DD1)
    //DD4     DD   DSNAME=SPEC52,DISP=(,KEEP),                        X
    //           SUBALLOC=(CYL,25,STEP1.DD1)
```

The data set from which space is to be allocated is defined on the DD statement named DD1 in STEP1. Fifty cylinders are allocated to the data set and the cylinders are contiguous. The DD statements named DD2, DD3, and DD4 in STEP2 request a portion of this space in the SUBALLOC parameter by referring the system to the data set defined on the DD statement named DD1 in STEP1. The order of the data sets on the volume, because of the request for suballocation, will be DUM, SPEC50, SPEC51, and SPEC52.

2.
```
//STEPX    EXEC PGM=GARV
//DD5      DD   DSNAME=SIMP,DISP=(,KEEP),UNIT=2311,                X
//             VOLUME=SER=315046,SPACE=(CYL,100,,CONTIG)
//DD6      DD   DSNAME=FIELD,DISP=(,KEEP),                         X
//             SUBALLOC=(1024,(800,60),DD5)
//STEPY    EXEC PGM=BERSS
//DD7      DD   DSNAME=PDS,DISP=(,KEEP),                           X
//             SUBALLOC=(CYL,(75,,8),STEPX.DD5)
```

The data set from which space is to be suballocated is defined on the DD statement named DD5 in STEPX. One hundred cylinders are allocated to the data set and the cylinders are contiguous. The DD statement named DD6 requests a portion of this space in units of blocks. The system computes how many tracks or cylinders are required for the data set. The DD statement named DD7 in STEPY also requests a portion of the space allocated to the data set defined on the DD statement named DD5 in STEPX. The DD statement named DD7 defines a partitioned data set and requests the system to allocate 8 256-byte records for a directory.

The SYSOUT Parameter

```
SYSOUT=(classname [,program name] [,form number])
                  [,
```

classname
    the class associated with the output device to which you want your output data set written.

,program name
    the member name of a program in the system library that is to write your output data set, instead of the system output writer, to a unit record device.

,
    specifies that the system output writer is to write your output data set to a unit record device, and a form number follows.

,form number
    specifies that the output data set should be printed or punched on a special output form.

Rules for Coding

1. The classname can be any alphameric character (A-Z, 0-9).

2. The form number is 1 to 4 alphameric and national (@,$,#) characters.

3. If a program name and form number are omitted, you need not enclose the classname in parentheses.

4. The UNIT, SPACE, OUTLIM, UCS, FCB, and DCB parameters can be coded with the SYSOUT parameter. Besides the mutually exclusive parameters listed below, other parameters coded with the SYSOUT parameter are ignored.

5. The DISP, DDNAME, AFF, SEP, VOLUME, LABEL, SPLIT, and SUBALLOC parameters and the SYSOUT parameter are mutually exclusive parameters; therefore, if any of these parameters are coded, do not code the SYSOUT parameter.

Advantages to Coding the SYSOUT Parameter

When you want a data set printed on an output listing or in the form of punched cards, you can code the UNIT parameter and identify the unit record device you want, or code the SYSOUT parameter and specify the class that corresponds to the type of unit record device you want. There are advantages to coding the SYSOUT parameter:

1. You can write your output data set to a direct access device and a system output writer writes the data set to a unit record device at a later time. This allows greater flexibility in scheduling print and punch operations, and improves operating system efficiency. You can also write your output data set directly to a unit record or magnetic tape device.

2. The output data set and system messages resulting from the job can be assigned to the same type of unit record device. This is accomplished by specifying the same classname in the SYSOUT and MSGCLASS parameters. (The MSGCLASS parameter is coded on the JOB statement.)

3. When you want the output data set printed or punched on a special output form, you can specify the form number in the SYSOUT parameter and let the system inform the operator at the time the data set is to be written what form is to be used.

THE CLASSNAME

When you code the SYSOUT parameter, you indicate a classname. A classname is an alphameric character (A-Z, 0-9) that corresponds to a type of unit record device. Each installation specifies what classnames correspond to what unit record devices. Therefore, when you specify a classname, the operator knows what type of unit record device you want and he ensures that a system output writer is available to write your output data set to the desired unit record device.

The system determines where system messages resulting from a job are to be written based on what is coded in the MSGCLASS parameter on the JOB statement. If the MSGCLASS parameter is not coded, system messages associated with your job are routed to the default output class specified in the PARM field of the input reader procedure. The default for the MSGCLASS parameter is A unless changed by your installation. Class A corresponds to a printer. If you want your output data set and the system messages resulting from the job written to the same unit record device, you simply code the same classname in both the MSGCLASS and SYSOUT parameters, or omit the MSGCLASS parameter and code your installation's default output class in the SYSOUT parameter.

THE PROGRAM NAME

The system provides system output writers, which transfer your output data set from a direct access volume to the desired unit record device. If there is a special installation program to handle this transfer, you can use this program, instead of a system output writer, by specifying the program's name as the second subparameter in the SYSOUT parameter. The program must be a member of the system library (SYS1.LINKLIB).

If you do not code a program name and code a form number as the last subparameter in the SYSOUT parameter, you must code a comma to indicate the absence of a program name.

THE FORM NUMBER

Each installation provides standard forms to contain printed or punched output. If there is a special output form you want to use, you can specify the form number as the last subparameter in the SYSOUT parameter. The system issues a message to the operator at the time the data set is to be printed or punched, which informs him of the form to be used. If you do not want system messages resulting from the job to appear on the special form, assign a classname in the MSGCLASS parameter on the JOB statement that is different from the classname assigned in the SYSOUT parameter.

CODING OTHER PARAMETERS WITH THE SYSOUT PARAMETER

The UNIT, SPACE, OUTLIM and DCB parameters can be coded with the SYSOUT parameter. The DDNAME, DISP, AFF, SEP, VOLUME, LABEL, SPLIT, UCS, and SUBALLOC parameters are mutually exclusive with the SYSOUT parameter; any other parameters that you code with the SYSOUT parameter are ignored.

You can write output data sets destined for unit records devices to a direct access device instead of immediately writing the data set to the desired unit record device. Later, a system output writer writes the data set to the desired unit record device. In the UNIT parameter, you can request what type of direct access device you want for writing the output data set, how many devices you want (up to a maximum of five), and unit separation from other data sets defined in the job step. In the SPACE parameter, you can specify how much space should be allocated to the data set and that unused space is to be released. If you omit the UNIT parameter, the system assigns a device; if you omit the SPACE parameter, the system assigns the amount of space to be allocated. These values are part of the PARM parameter field in the input reader procedure used to read the input stream.

You can also write an output data set directly to the desired unit record or magnetic tape device. When direct system output is desired, the operator selects a unit record or magnetic tape device for a class by issuing a START DSO (direct system output) command. In addition to the SYSOUT parameter, the DCB and UCS parameters can be coded. If the SYSOUT subparameters other than classname are coded, the specified information is ignored. The UNIT and SPACE parameters are also ignored if direct system output processing is used. Since the type of processing to be used may not always be known, it is advisable to code these parameters in case an intermediate direct access device is used.

The DCB parameter can be coded with the SYSOUT parameter to complete the data control block associated with the output data set. The information contained in this data control block is used when the data set is written to the direct access device and read by the system output writer. However, the output writer's own DCB attributes are used when the data set is written to the desired unit record device.

The OUTLIM parameter allows you to specify a limit for the number of logical records you want included in the output data set being routed through the output stream. The OUTLIM parameter has meaning only in systems with the System Management Facilities option with system, job, and step data collection. Unless the SYSOUT parameter is coded in the operand field of the same DD statement, the OUTLIM parameter is ignored.

JOB SEPARATORS

Your output data is preceded by a job separator if your installation incorporated routines to write job separators. A job separator is a series of three listing pages or three punched cards that separates the output data sets of different jobs. The output data sets from these jobs were written to the same unit. Each page or card contains the name of the job whose data follows, and identifies the output class. Job separators make it easier for the operator to separate the data produced by your job from the data of other jobs.

Examples of the SYSOUT Parameter

1.
```
//DD1     DD   SYSOUT=P
```

This DD statement specifies that the data set is to be written to the unit record device corresponding to class P. Since the UNIT and SPACE parameters are not coded, the system obtains device and space allocation information from the input reader procedure.

2.
```
//JOB50   JOB  ,'R.J. WALKER',MSGCLASS=C
//STEP1   EXEC PGM=SET
//DDX     DD   SYSOUT=C,DCB=(BUFNO=4,OPTCD=W)
```

The DD statement named DDX specifies that the data set is to be written to the unit record device corresponding to class C. The DCB parameter is coded to complete the data control block associated with this data set. Since the classnames in the SYSOUT parameter and the MSGCLASS parameter, on the JOB statement, are the same, the system messages resulting from this job and the output data set are written to the same unit record device.

3. `//DD5  DD  SYSOUT=A,UNIT=2314,SPACE=(CYL,(12,1),RLSE)`

This DD statement specifies that the data set is to be written to the unit record device corresponding to the standard output class A. The system assigns a 2314 unit and allocates 12 cylinders to the data set, rather than obtaining device and space allocation information from the input reader procedure. Since the RLSE subparameter is coded in the SPACE parameter, any unused space is released.

4. `//DD6  DD  SYSOUT=(F,,7402)`

This DD statement specifies that the data set is to be written to the unit record device corresponding to class F and the output data set is to be printed on a special form. The form number is 7402.

5. `//DD7  DD  DUMMY,SYSOUT=A,DCB=(BLKSIZE=800,LRECL=400,RECFM=FB)`

This DD statement specifies that no writing or allocation is to be done for the data set, but the specified information is to placed in the appropriate system control blocks.

The TERM Parameter -- MVT and TSO

```
TERM=TS
```

TS
indicates to the system that the input or output data being defined is coming from or going to a time sharing terminal.

Rules for Coding

1. TERM=TS is effective only under the operating system with MVT and the Time Sharing Option (TSO). The TERM parameter is ignored in batch processing, in an MFT operating system, or in a system without TSO.

2. TS is the only value that can be specified by the TERM parameter. If any other value is used, a JCL error message is produced.

3. A DD statement with TERM=TS can only be concatenated if it is the last DD statement.

4. Except for the DCB parameter, all other parameters (including DUMMY, DYNAM and DSNAME=NULLFILE) coded on a DD statement with TERM are ignored.

5. If the TERM parameter is coded for batch processing, the parameter is not used, but is checked for syntax.

6. The TERM and DDNAME parameters are mutually exclusive; the TERM and DYNAM parameters are mutually exclusive. These combinations should not be coded.

What the TERM Parameter Does

The TERM parameter notifies the operating system that the data set (represented by the DD statement that contains the TERM parameter) is coming from or going to a time sharing terminal. TERM allows your time sharing job to communicate with a terminal device. For example, your program can put out messages to a terminal user requesting data input records. Your program can then read in the data input records supplied by the terminal user, perform operations with this data, and then put out the results to the terminal.

Example of the TERM Parameter

1.
```
//DD1     DD   TERM=TS
or
//DD2     DD   UNIT=2400,DISP=(MOD,PASS),TERM=TS
```

The above two DD statements are equivalent in effect. In the time sharing environment, all the parameters coded on the second DD statement are ignored except the TERM parameter. In a batch processing environment, the UNIT and DISP parameters are used but TERM is ignored.

2.
```
//DD3     DD   UNIT=2400,DISP=(MOD,PASS),DCB=(LRECL=80,BLKSIZE=80),
             TERM=TS,LABEL=(,NL)
```

In a time sharing environment, all the parameters in the above example except TERM and DCB are ignored.

The UCS Parameter

```
UCS=(character set code [,FOLD]    [,VERIFY])
                        [,    ]
```

character set code
identifies the special character set you want for printing the data set.

,FOLD
specifies that you want the chain or train corresponding to the desired character set loaded in the fold mode.

,
specifies that the chain or train is not to be loaded in the fold mode and the VERIFY subparameter follows.

,VERIFY
specifies that the operator is to verify that the correct chain or train is mounted before the data set is printed.

Rules for Coding

1. The character set code can be 1 through 4 characters.

2. If the FOLD and VERIFY subparameters are omitted, you need not enclose the character set code in parentheses.

3. If the UCS parameter is coded and the data set is not written to a printer with the universal character set (UCS) feature, the UCS parameter is ignored.

4. The UCS and DDNAME parameters and the DCB subparameters RKP, CYLOFL, and INTVL are mutually exclusive parameters; therefore, if the DDNAME parameter or one of the DCB subparameters RKP, CYLOFL, or INTVL is coded, do not code the UCS parameter.

5. If the SYSOUT parameter is coded and an asynchronous writer is used, the UCS parameter is only recognized for the 3211 printer.

Special Character Sets

The Universal Character Set (UCS) feature allows you to alternately use different sets of print characters. It is available as a special feature on the 1403 printer and as a standard feature on the 3211 printer.

In the UCS parameter you specify what character set you want to use; the operator ensures that the corresponding chain or train is mounted on the printer. In order to use a particular special character set, an image of the character set must be contained in SYS1.IMAGELIB and the chain or train corresponding to the character set must be available for use. IBM provides standard special character sets and the installation may provide user-designed special character sets. How to include the images for these special character sets in SYS1.IMAGELIB is discussed in Data Management for System Programmers.

If you omit the UCS parameter and the data set is written to a printer with the UCS feature, a default character set is used. If the chain or train mounted on the printer does not correspond to a default character set, the operator is requested to identify a default character set and mount the corresponding chain or train.

Note: When the UCS parameter is coded with the SYSOUT parameter and the data set is first written to a direct access device, the UCS specification is not kept. In this case, the operator must either specify the UCS parameter in the START WTR command or add the UCS parameter to the user-designed output writer procedure, if the data set is to be printed with the desired character set.

The first subparameter of the UCS parameter identifies the character set you want for printing your data set. Each character set has a unique 1- through 4-byte code.

IDENTIFYING THE CHARACTER SET

The codes for the IBM standard special character sets are:

Codes for 1403	Codes for 3211	Characteristics
AN	A11	Arrangement A, standard ECBDIC character set. 48 characters.
HN	H11	Arrangement H, EBCDIC character set for FORTRAN and COBOL, 48 characters.
	G11	ASCII character set.
PCAN		Preferred alphameric character set, arrangement A.
PCHN		Preferred alphameric character set, arrangement H.
PN	P11	PL/1 alphameric character set.
QN		PL/1 preferred alphameric character set for scientific applications.
QNC		PL/1 preferred alphameric character set for commercial applications.
RN		Preferred character set for commercial applications of FORTRAN and COBOL.
SN		Preferred character set for text printing.
TN	T11	Character set for text printing, 120 characters.
XN		High-speed alphameric character set for 1403, Model 2.
YN		High-speed preferred alphameric character set for 1403, Model 3 or N1.

For each user-designed special character set, the installation assigns a unique code. If you want to use one of these, specify the corresponding code in the UCS parameter.

REQUESTING FOLD MODE

FOLD can be coded as the second subparameter of the UCS parameter and requests the fold mode. The fold mode is described in the publication IBM 2821 Control Unit, GA24-3112. The fold mode is most often requested when uppercase and lowercase data is to be printed only in uppercase.

The FOLD subparameter is a positional subparameter. If you omit the FOLD subparameter and code the VERIFY subparameter, you must code a comma to indicate the absence of FOLD.

REQUESTING OPERATOR VERIFICATION

VERIFY can be coded as the last subparameter of the UCS parmeter and requests that the operator visually verify that the character set image corresponds to the graphics of the chain or train that was mounted. When VERIFY is coded, the character set image is displayed on the printer so that the operator can make the verification before the data set is printed.

Examples of the UCS Parameter

1. `//DD1 DD UNIT=1403,UCS=(YN,,VERIFY)`

   This DD statement defines an output data set that is to be written to a 1403 printer. The UCS parameter requests that the data set be written using the chain or train corresponding to the special character set with the code YN. Since VERIFY is coded, the character set image is displayed on the printer before the data set is printed.

2. `//DD2 DD SYSOUT=G,UCS=PCHN`

   This DD statement defines an output data set that is to be written to the unit record device that corresponds with class G. If the device is a printer with the universal character set and the writer is a direct SYSOUT writer, the request in the UCS parameter for the special character set with the code PCHN is recognized. Otherwise, the UCS parameter is ignored.

The UNIT Parameter

```
{UNIT=(  [unit address]  [,unit count]  [,DEFER]  [,SEP=(ddname,...)])}
        [device type    ]  [,P         ]
        [group name     ]  [,          ]
{UNIT=AFF=ddname                                                     }
```

unit address
: identifies a particular unit by its address, which consists of the channel, control unit, and unit numbers.

device type
: identifies a particular type of device.

group name
: identifies a particular group of devices. The group name and the devices that make up a group are specified during system generation.

,unit count
: indicates the number of devices you want assigned to the data set.

,P
: specifies that each volume on which the data set resides is to be assigned a device.

,
: specifies that only one device is required and another subparameter follows. (If the DEFER subparameter is not coded but the SEP parameter is coded, this comma is optional.)

,DEFER
: specifies that the system should assign a device(s) to the data set but the volume(s) on which the data set resides should not be mounted until the data set is opened.

,SEP=
: indicates that this data set is to be assigned a different direct access device than the devices assigned to certain other data sets, i.e., unit separation.

(ddname,...)
: the names of up to eight earlier DD statements in the job step that define data sets from which you want unit separation.

AFF=
: indicates that the system should assign the data set to the same device(s) as assigned to another data set, i.e., unit affinity.

ddname
: the name of an earlier DD statement in the job step that defines a data set with which you want unit affinity.

Rules for Coding

1. If the only subparameter coded in the UNIT parameter is the first subparameter, you need not enclose it in parenthses.

2. If the SEP subparameter is the only subparameter you are coding in the UNIT parameter, code UNIT=(,SEP=(ddname,...)).

3. If the list of ddnames consists of only one ddname, you need not enclose it in parentheses.

4. You need not code the unit count subparameter if you want only one device assigned to the data set.

5. The UNIT and DDNAME parameters are mutually exclusive parameters; therefore, if DDNAME is coded, do not code the UNIT parameter.

Providing Unit Information

Before the data set can be used as input to a processing program or written as output by a processing program, the volume on which a data set resides or will reside must be mounted on an input/output device. The UNIT parameter provides the system with the information it needs to assign a device to the data set.

In order for the system to assign a device, you must provide in the UNIT parameter:

- The specific unit you want: code the unit address, or a general description of the device; code the device type or group name.

Optionally, you can:

- Specify how many devices you want assigned to the data set when more than one device is required. You can code the unit count and specify how many devices are required, or in certain cases, imply how many devices are required by coding P.

- Request the system to assign a device to a data set and not to cause the volume on which the data set resides to be mounted until the data set is opened.

- Request the system to assign a data set to a device other than the devices assigned to data sets defined in the same job step; code the keyword subparameter SEP and identify the data sets from which you want unit separation.

Another way to provide unit information is to request unit affinity with another data set by coding UNIT=AFF=ddname. The system obtains unit information from the named DD statement.

Except in a few cases, the UNIT parameter is always coded on a DD statement that defines a data set that requires one or more devices. In the following cases, the system obtains the required unit information from other sources. Therefore, you need not code the UNIT parameter:

- When the data set is cataloged. For cataloged data sets, the system obtains unit and volume information from the catalog. However, if VOLUME=SER=serial number is coded on a DD statement that defines a cataloged data set, the system does not look in the catalog. In this case, you must code the UNIT parameter. If the VOLUME parameter is not coded but you request a device in the UNIT parameter, the request is ignored.

- When the data set is passed from a previous job step. For passed data sets, the system obtains unit and volume information from an internal table. However, if VOLUME=SER=serial number is coded on a DD statement that defines a passed data set, the system does not look in the internal table. In this case, you must code the UNIT parameter. If the VOLUME parameter is not coded but you request a device in the UNIT parameter, the request is ignored.

- When the data set is to use the same volumes assigned to an earlier data set, i.e., VOLUME=REF=reference is coded. In this case, the system obtains unit and volume information from the earlier DD statement that specified the volume serial number or from the catalog. If you request a device in the UNIT parameter, the request is ignored.

- When the data set is to share space or cylinders with an earlier data set, i.e., SUBALLOC or SPLIT is coded. In this case, the system obtains unit and volume information from the earlier DD statement that specifies the total amount of space required for all the data sets. If the VOLUME parameter is coded, it is ignored. If you request a device in the UNIT parameter, the request is ignored. When using VOL=REF reference to a previous DD statement that uses esoteric names the system will allocate to generic, rather than esoteric, names.

In all of these cases, you can code the UNIT parameter when you want more devices assigned.

IDENTIFYING THE DEVICE

You must identify to the system the specific device you want or the type of device you want. To identify a specific device, you must specify a unit address. When a unit address is coded, the system assigns you that unit.

There are two ways to identify the type of device you want: specify a device type, which corresponds to a particular set of device features, or specify a group name, which identifies a group of devices that may be different models. When a device type is coded, the system assigns an available device of that type. When a group name is coded, the system assigns an available device that is part of that group. In all cases, the block size specified for the data cannot exceed the maximum block size permitted for the assigned device.

Unit Address

To identify a device by its unit address, you specify the 3-byte address of the unit. The address is made up of the channel, control unit, and unit numbers. For example, UNIT=180 indicates you want channel 1, control unit 8, and unit 0.

To request a specific bin on a specific 2321, you should code UNIT=address/bin, where "bin" is a number from 0 through 9. For example, UNIT=293/5 indicates you want channel 2, control unit 9, device 3, and bin 5. If you code UNIT=293, you are requesting one of the available bins on that unit.

If you identify a telecommunications device by its unit address, the system will allocate that device on a shared basis whether or not the device is already allocated. Offline telecommunications devices will be allocated just as previously allocated ones will be.

You should not identify a device by its address unless it is absolutely necessary. Specifying a unit address limits unit assignment and may result in a delay of the job if the unit is being used by another job.

Device Type

Device types correspond to particular set of features of input/output devices. When you code a device type, you allow the system to assign any available device of that device type. For example, if the device type

you want is a 2302 Disk Storage Drive, you code UNIT=2302. The system assigns an available 2302. If only one device in the system is of that device type, the system assigns that device. If there is more than one device in the system of that device type, there is a certain degree of device independence.

The device types that can be coded and their descriptions are listed below. (You can code only those device types that were defined during system generation.)

TAPE

Device Type	Device
2400	2400 series Nine-Track Magnetic Tape Drive that can be allocated to a data set written or to be written in 800 bpi when the dual-density feature is not installed on the drive or in 1600 bpi when the dual-density feature is installed on the drive.
2400-1	2400 series Magnetic Tape Drive with Seven-Track Compatibility and without Data Conversion.
2400-2	2400 series Magnetic Tape Drive with Seven-Track Compatibility and Data Conversion.
2400-3	2400 series Nine-Track Magnetic Tape Drive that can be allocated to a data set written or to be written in 1600 bpi density.
2400-4	2400 series Nine-Track Magnetic Tape Drive having an 800 and 1600 bpi density capability.
3400-1	3410 Magnetic Tape with Seven-track Compatability and Data Conversion that can be allocated to a data set written in 200, 556, or 800 bpi density.
3400-2	3420 Magnetic Tape with Seven-Track Compatibility and Data Conversion that can be allocated to a data set written in 556 or 800 bpi density.
3400-3	3410 or 3420 Nine-Track Magnetic Tape drive that can be allocated to a data set written or to be written in 1600 bpi density.
3400-4	3410 or 3420 Nine-Track Magnetic Tape drive having an 800 and 1600 bpi density capability.

The system may assign a tape drive from the 3400 series to satisfy a request for a 2400-series tape. However, the system will not assign a 2400-series tape to satisfy a request for a 3400 tape drive.

DIRECT ACCESS

Device Type	Device
2301	2301 Drum Storage Unit.
2302	2302 Disk Storage Drive.
2303	2303 Drum Storage Unit.
2305-1	2305 Fixed Head Storage Facility Model 1
2305-2	2305 Fixed Head Storage Facility Model 2
2311	2311 Disk Storage Drive.
2314	2314 Storage Facility.
2319[1]	2319 Disk Storage Facility
2321	any bin mounted on a 2321 data cell drive.
3330	3330 Disk Storage Drive

[1]To indicate the 2319 in the UNIT parameter, specify UNIT=2314. However, to designate the 2319 as the particular device for your data set, specify UNIT=unit address.

UNIT RECORD

Device Type	Device
1052	1052 Printer-Keyboard.
1275	1275 Optical Reader Sorter (available through Word Trade branch offices only)
1285	1285 Optical Reader
1287	1287 Optical Reader
1288	1288 Optical Reader
1403	1403 Printer or 1404 Printer (continuous form only).
1419	1419 Magnetic Character Reader
1442	1442 Card Read Punch.
1443	1443 Printer.
2495	2495 Tape Cartridge Reader
2501	2501 Card Reader.
2520	2520 Card Read Punch.
2540	2540 Card Read Punch (read feed).
2540-2	2540 Card Read Punch (punch feed).
2671	2671 Paper Tape Reader.
3210	3210 Printer-Keyboard

3211	3211 Printer
3215	3215 Printer-Keyboard
3505	Card Reader
3525	Card Punch (read and print features)

GRAPHIC

Device Type	Device
1053	1053 Model 4 Printer.
2250-1	2250 Display Unit, Model 1.
2250-3	2250 Display Unit, Model 3.
2260-1	2260 Model 1 Display Station (Local Attachment).
2260-2	2260 Model 2 Display Station (Local Attachment).
2280	2280 Film Recorder.
2282	2282 Film Recorder/Scanner.
3277-1	3277 Display Station (Model 1), Local Attachment
3277-2	3277 Display Station (Model 2), Local Attachment
3284-1	3284 Printer (Model 1), Local Attachment
3284-2	3284 Printer (Model 2), Local Attachment
3286-1	3286 Printer (Model 1), Local Attachment
3286-2	3286 Printer (Model 2), Local Attachment

Group Name

A group name is 1 through 8 alphameric characters and identifies a device or a group of devices. The group of devices can consist of devices of the same type or different direct access and tape device types. Group names are established during system generation.

When you code a group name, you allow the system to assign any available device that is included in the group. (If a group consists of only one device, the system assigns that device.) For example, if all 2301 and 2303 Drum Storage Units are included in the group named DRUM and you code UNIT=DRUM, the system assigns an available 2301 or 2303 device.

If extending a data set that was created using the group name parameter, the additional units allocated will be of the same device type. They will not necessarily be of that same group.

A group may consist of more than one device type. In this case, you should not code this group's group name when you are defining an existing data set, since the volume(s) on which the data set resides may require a different device than the one assigned by the system, i.e., a tape volume must be assigned to a tape device, not a direct access device.

When creating a data set which could require more than one device, a group name which includes more than one device type should not be used. This is necessary to prevent multiple device types from being allocated to the request.

When the automatic volume recognition feature is included in the system and you specify a group name, this feature will assign devices to volumes already mounted, but will not request mounting of any volume that is not mounted.

UNIT COUNT

The unit count subparameter indicates how many devices you want assigned to a data set. If you do not code this subparameter (or code 0) and you do not request parallel mounting, the system assumes you are requesting one device. If you receive a passed data set or refer the system to a cataloged data set or earlier DD statement for volume and unit information (VOLUME=REF=reference), the system will also assume a unit count of 1, even if more devices were requested in an earlier DD statement. The system may ignore a request for a specific number of units if the data set has volume affinity with at least one other data set. See the description of volume affinity in the section on the VOLUME parameter for specific details.

For operating efficiency, you can request multiple devices for a multivolume data set or for a data set that may require additional volumes. When each required volume is mounted on a separate device, time is not lost during execution of the job step while the operator demounts and mounts volumes. The maximum number of devices that can be requested per DD statement is 59.

In the following cases, you should always code the unit count subparameter when the data set may be extended to a new volume:

- If the data set resides on a permanently resident or reserved volume. In these two cases, the volume cannot be demounted in order to mount another volume.
- If the data set is assigned space through suballocation. Code the unit count subparameter on the DD statement that requests the space to be suballocated.

The unit count subparameter is a positional subparameter, and it shares the same position as the subparameter P. If neither of these subparameters is coded and the DEFER or SEP subparameter follows, code a comma to indicate the absence of the unit count subparameter and the subparameter P. (If the DEFER subparameter is not coded but the SEP parameter is coded, you may omit the comma.)

PARALLEL MOUNTING

Requesting parallel mounting has the same effect as specifying a unit count, i.e., more than one device is assigned to the data set. When parallel mounting is requested, the system counts the number of volume serial numbers specified on the DD statement and assigns to the data set as many devices as there are serial numbers. (For cataloged data sets, the system counts the number of volume serial numbers contained in the catalog.) You request parallel mounting by coding the letter P in place of the unit count subparameter.

The subparameter P is a positional subparameter, and it shares the same position as the unit count subparameter. If neither of these subparameters is coded and the DEFER or SEP subparameter follows, code a comma to indicate the absence of the subparameter P and the unit count subparameter. (If the DEFER subparameter is not coded but the SEP subparameter is coded, you may omit the comma.)

DEFERRED MOUNTING

The DEFER subparameter requests the system to assign the required units to a data set and to defer the mounting of the volume(s) on which the data set resides until the processing program attempts to open the data set. The DEFER subparameter should only be coded on DD statements that define data sets residing on removable volumes. The DEFER subparameter cannot be coded on a DD statement that defines an indexed sequential data set or that defines a new data set that is to be written on a direct access volume, because space cannot be allocated to the data set. If DEFER is coded on a DD statement that defines a new direct access data set, DEFER will be ignored.

If you request deferred mounting of a volume and the data set on that volume is never opened by the processing program, the volume is not mounted during the execution of the job step. If a later job step refers to that data set, the system may assign a different device to the data set than was originally assigned to it.

UNIT SEPARATION

When you make nonspecific volume requests for data sets defined in a job step, the system assigns volumes to the data sets. If the DD statements that define these data sets request the same type of device, the system may assign more than one data set to the same device.

If you do not want a data set to be assigned to the same device that is assigned to other data sets, you can request this in the SEP subparameter. A request for unit separation has meaning only for direct access devices.

The SEP subparameter appears as the last subparameter in the UNIT parameter. To identify the data sets that should not be assigned the same device as this data set, follow SEP= with a list of ddnames of the DD statements that define these data sets. The listed DD statements must precede this statement and must be contained in the same job step. The list of ddnames must be enclosed in parentheses, unless there is only one ddname. If one of the listed DD statements defines a dummy data set, the system ignores the unit separation request for that data set.

The number of separations allowed per DD statement is the number of implied separations plus the number of specified separations to a maximum of eight. Each time a DD name is the object of a separation request, it assumes an implied separation from the requesting DD statement. Thus, if a DD statement was referenced in the SEP parameter in a following DD statement, the referenced DD statement would be allowed to specify up to seven separations.

When you make a specific volume request for a data set and request unit separation for that data set, the system issues a message to the operator if the request for unit separation cannot be satisfied. The operator decides if the system should wait for devices to become available, or if the request for unit separation should be ignored, or if the job should be cancelled. When you make a nonspecific volume request for a data set and request unit separation for that data set, the request may be ignored, depending on how many disk drives are available and how much space is available on those disk drives. If

there is not a sufficient number of devices to satisfy the separation request, it will be ignored and no message will be issued. If the separation request cannot be satisfied because of insufficient space on a volume(s), a message will be issued giving the operator the option to wait for available devices, cancel the job, or ignore unit separation.

Unit Affinity

To conserve the number of devices used in a job step, you can request that an existing data set be assigned to the same device or devices as assigned to a data set defined earlier in the job step. When two data sets are assigned the same device, the data sets are said to have unit affinity. When the data sets reside on different volumes, unit affinity implies deferred mounting for one of the volumes, since both volumes cannot be mounted on the same device at the same time. Unit affinity is invalid if coded for a new direct access data set, except when an explicit volume reference is made. When a volume which is assigned to that device becomes private, all subsequent volumes assigned to that device through UNIT=AFF=ddname also become private.

You request unit affinity by coding UNIT=AFF=ddname on a DD statement. The ddname is the name of an earlier DD statement in the same job step, and the system obtains unit information from this statement. The data set defined on the DD statement that requests unit affinity is assigned the same device or devices as the data set defined on the named DD statement. If the ddname refers to a DD statement that defines a dummy data set, the data set defined on the DD statement requesting unit affinity is assigned a dummy status.

When unit affinity is requested for two data sets that reside on different 2321 volumes, the data sets are assigned the same device but may be assigned different bins. If the data sets are assigned different bins, the implied deferred mounting is ignored.

Examples of the UNIT Parameter

1.
```
//DD1      DD   DSNAME=AAG3,DISP=(,KEEP),                          X
//             VOLUME=SER=13230,UNIT=2400
```

This DD statement defines a new data set and requests the system to assign any 2400 9-track tape drive to the data set. If a 2400 9-track tape drive is not available, the system will assign a 3400-4 type tape drive.

2.
```
//DYD      DD   DSNAME=AAG4,DISP=OLD,VOLUME=SER=12345,UNIT=3400-3
```

This DD statement defines an existing data set that resides on a tape volume and requests the system to assign a 3400 9-track 1600 bpi tape device to the data set.

3.
```
//DD2      DD   DSNAME=X.Y.Z,DISP=OLD,UNIT=(,2)
```

This DD statement defines a cataloged data set and requests the system to assign two devices to the data set. The device type is obtained from the catalog.

4.
```
//DD3      DD   DSNAME=COLLECT,DISP=OLD,                          X
//             VOLUME=SER=1095,UNIT=(DISK,,DEFER)
```

This DD statement defines an existing data set that resides on a direct access volume and requests the system to assign any device that is part of the group named DISK. Since DEFER is coded, the volume is not mounted until the data set is opened.

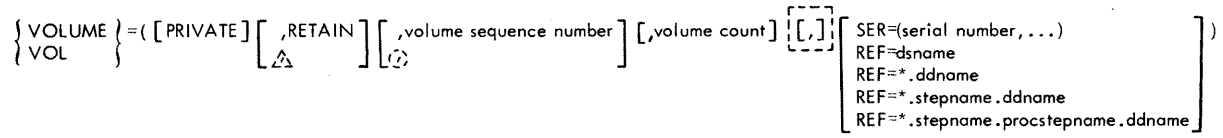
5.
```
//STEP1   EXEC PGM=XTRA
//DDA     DD   UNIT=2311,SPACE=(1024,(150,20))
//DDB     DD   UNIT=2311,SPACE=(1024,(100,10))
//DDC     DD   UNIT=(2311,SEP=(DDA,DDB)),SPACE=(2048,(300,30))
```

The DD statements in this job step define temporary data sets. The DD statement named DDC requests the system to assign the data set to a different device than is assigned to either of the data sets defined on the DD statements named DDA and DDB.

6.
```
//STEP2   EXEC PGM=POINT
//DDX     DD   DSNAME=EST,DISP=MOD,VOLUME=SER=(42569,42570),     X
//            UNIT=(2311,2)
//DDY     DD   DSNAME=ERAS,DISP=OLD,UNIT=2400-2
//DDZ     DD   DSNAME=RECK,DISP=OLD,                             X
//            VOLUME=SER=(40653,13262),UNIT=AFF=DDX
```

The DD statement named DDZ requests that the system assign the same unit to this data set as it assigns to the data set defined on the statement named DDX. Since DDX requests two devices, these two devices are assigned to the data set defined on DDZ.

The VOLUME Parameter



PRIVATE
indicates that no output data set can be allocated to this volume unless the volume is specifically requested, and the volume is to be demounted after its last use in the job step, unless RETAIN is coded or the data set is passed.

,RETAIN
indicates that this volume is not to be demounted after its last use in the job step.


indicates that the volume need not be retained after the job step and the volume sequence number or volume count subparameter follows.

,volume sequence number
specifies which volume of an existing multivolume data set you want to begin processing with.


indicates that you want to begin processing of an existing multivolume data set with the first volume, and the volume count subparameter follows.

,volume count
specifies the maximum number of volumes an output data set requires.


specifies that either the SER or REF subparameter follows and one or more subparameters precede it.

SER=
indicates that the serial numbers of the volumes on which the data set resides or will reside follow.

(serial number,...)
the serial numbers of the volumes on which the data set resides or will reside.

REF=
indicates that the serial numbers of the volumes on which the data set resides or will reside are identified on an earlier DD statement in the job or in the catalog.

dsname
the name of a cataloged or passed data set. The system locates the information about the data set and assigns your data set to the same volumes as are assigned to the cataloged or passed data set.

*.ddname
specifies that the system must obtain the volume serial numbers from an earlier DD statement named "ddname" in the same job step.

*.stepname.ddname
specifies that the system must obtain the volume serial numbers from a DD statement named "ddname," which was defined in an earlier job step named "stepname."

*.stepname.procstepname.ddname
specifies that the system must obtain the volume serial numbers from a DD statement named "ddname," which was defined in an earlier procedure step named "procstepname"; the procedure step is part of a procedure that was called by an earlier job step named "stepname."

Rules for Coding

1. The volume sequence number subparameter can be 1 to 3 digits.

2. The volume count subparameter is a number from 1 through 255.

3. If the only subparameter you are coding is PRIVATE, you need not enclose it in parentheses.

4. If the only subparameter you are coding is SER or REF, code VOLUME=SER=(serial number,...) or VOLUME=REF=reference.

5. If the list of volume serial numbers consists of only one serial number, you need not enclose the serial number in parentheses.

6. The VOLUME, DDNAME, and SYSOUT parameters are mutually exclusive parameters; therefore, if DDNAME or SYSOUT is coded, do not code the VOLUME parameter.

7. When coding a volume reference to a direct access data set defined in the same step, and the data set status of the referencing DD statement is old, the referenced DD statement must contain a specific volume request.

Providing Volume Information

A volume can be a tape reel, a disk pack, a data cell, a drum, or part of an IBM 2302 Disk Storage device served by one access mechanism. The VOLUME parameter provides information about the volume or volumes on which an input data set resides or on which an output data set will reside.

Before a data set can be read or written, the volume on which the data set resides or will reside must be mounted. For an existing data set, you must identify the volume or volumes on which the data set resides by making a specific volume request. For a new data set, you can make a specific volume request or let the system select a volume for you by making a nonspecific volume request.

Specific Volume Request

A specific volume request informs the system of the volume's serial number. Any of the following implies a specific volume request:

1. The data set is passed from an earlier step or is cataloged.

2. VOLUME=SER=serial number is coded on the DD statement.

3. VOLUME=REF=reference is coded on the DD statement, referring to an earlier specific volume request.

When you make a specific volume request, you can code the PRIVATE subparameter or the PRIVATE and RETAIN subparameters in the VOLUME parameter. For passed data sets, you can also code the volume count subparameter. For cataloged data sets, you can also code the sequence number and volume count subparameters.

Nonspecific Volume Request

A nonspecific volume request can be made only if you are defining a new data set. When you make a nonspecific volume request, the system may assign your data set to a volume that is already mounted or may cause a volume to be mounted. What the system does depends on the volume state of the volumes that are already mounted. The volume states that mounted volumes can assume and how they affect volume selection are described under "Volume States" at the end of this chapter.

When you make a nonspecific volume request, you can code the PRIVATE subparameter, or the PRIVATE and RETAIN subparameters, and the volume count subparameter in the VOLUME parameter. You should not code the volume sequence number subparameter when you make a nonspecific volume request.

THE PRIVATE SUBPARAMETER

When you make a specific or nonspecific volume request, you can code PRIVATE as the first subparameter in the VOLUME parameter. The volume assigned is called a private volume. This private volume cannot then be assigned to any other data set for which a nonspecific volume request is made. In addition, a private volume is demounted after its last use in the job step unless RETAIN or PASS is coded or the volume is a permanently resident or reserved volume. (Permanently resident and reserved volumes are described under "Volume States" at the end of this chapter.)

If PRIVATE is the only subparameter coded in the VOLUME parameter, you need not enclose it in parentheses.

When PRIVATE Is Not Coded

What occurs when PRIVATE is not coded depends on the type of volume request and whether a direct access or tape device is requested.

Specific request for a direct access volume: If PRIVATE is not coded and you make a specific request for a direct access volume, the volume assigned is called a public volume. A public volume remains mounted after its last use in a step so that it can be used again without the need to remount it.

Nonspecific request for a direct access volume: If PRIVATE is not coded and you make a nonspecific request for a direct access volume and the data set is temporary, the system assigns a volume called a public volume. If PRIVATE is not coded and you make a nonspecific request for a direct access volume and the data set is nontemporary, the system assigns a volume called a storage volume. Public and storage volumes remain mounted after their last use in a step so that they can be used again without the need to remount them. If it is possible that the data set may require more space than was requested for it, request more than one volume in the volume count subparameter of the VOLUME parameter and more than one device in the unit count subparameter of the UNIT parameter.

Specific request for a tape volume: If PRIVATE is not coded and you make a specific request for a tape volume, the system treats it as a request for a private volume. (How this affects the volume is described in the previous topic "The PRIVATE Subparameter.")

Nonspecific request for a tape volume: If PRIVATE is not coded and you make a nonspecific request for a tape volume and the data set is nontemporary, the system treats it as a request for a private volume. (As mentioned earlier, the system always considers certain requests to

be specific. For tape volumes, the system also considers the following to be a specific request: a status of OLD or SHR and a disposition of other than DELETE coded in the DISP parameter.) How a request for a private volume affects the volume is described in the previous topic "The PRIVATE Subparameter."

If PRIVATE is not coded and you make a nonspecific request for a tape volume and the data set is temporary, the system assigns a volume called a scratch volume. A scratch volume remains mounted after its last use in a step so that it can be assigned again without the need to remount it. If it is possible that the data set may exceed one volume, request more than one volume in the volume count subparameter of the VOLUME parameter and more than one device in the unit count subparameter of the UNIT parameter.

When PRIVATE is not coded, and the volume sequence number or volume count subparameter is coded, you must code a comma to indicate the absence of PRIVATE.

THE RETAIN SUBPARAMETER

If you have coded PRIVATE as the first subparameter in the VOLUME parameter, you may want to code RETAIN as the second subparameter. RETAIN overrides the system action of demounting a private volume after its use in a job step. For a tape volume, the volume remains mounted until after it is used in a subsequent step or until the end of the job, whichever occurs first. For volumes other than tape, the volume remains mounted until the end of the job. If the data set resides on more than one volume and the volumes are mounted in sequential order, only the last volume is retained.

The RETAIN subparameter need not be coded when the data set is to be passed; the system automatically retains the volumes on which the data set resides.

If the RETAIN subparameter is not coded and the volume sequence number or volume count subparameter follows, code a comma to indicate the absence of RETAIN.

THE VOLUME SEQUENCE NUMBER SUBPARAMETER

When you are reading or lengthening an existing multivolume data set, you can begin processing with other than the first volume of the data set by coding a volume sequence number. The sequence number must be less than or equal to the number of volumes on which the data set exists and can range from 1 to 4 digits. A volume sequence number is normally coded when volume serial numbers are not specified on the DD statement (i.e., you are retrieving a cataloged data set or VOLUME=(,,seq#,REF=reference) is coded). If both a volume sequence number and volume serial numbers are coded in the VOLUME parameter, you will begin processing with the volume that corresponds with the volume sequence number.

The volume sequence number is a positional subparameter and must follow the PRIVATE and RETAIN subparameters or the commas that indicate their absence. If the volume sequence number subparameter is not coded and the volume count subparameter follows, code a comma to indicate the absence of a sequence number.

If a volume sequence number is used with a nonspecific volume request, the results are unpredictable.

THE VOLUME COUNT SUBPARAMETER

The volume count subparameter tells the system the maximum number of volumes an output data set may require. If a volume count of 1 to 5 is specified, the maximum number is five; if a count of 6 to 15 is specified, the maximum number is 15. If a volume count above 15 is specified, the actual maximum number allowed will be a multiple of 15 up to an absolute maximum of 255. The total number for all the DD statements in one job step cannot exceed 4095.

When you make a nonspecific volume request and the data set may exceed one volume, request more than one volume in the volume count subparameter and code PRIVATE or request the same number of devices as volumes. If the volume(s) you initially specified is demountable, the system will request scratch volumes to be mounted until either the data set is complete or all entries in the JFCB are filled. If the JFCB entries are already filled, or the volumes are not demountable, the job step will abnormally terminate. When you request a non-specific tape volume for a data set with no labels, the system assigns the volume serial numbers required for the data set. If a volume count greater than 99 is specified, duplicate volume serial numbers are assigned.

When you make a specific volume request and the data set may require use of more volumes than there are serial numbers, specify in the volume count subparameter the total number of volumes that may be used. By requesting multiple volumes in the volume count subparameter, you can ensure that the data set can be written on more than one volume if it exceeds one volume.

If you make a nonspecific volume request and the volume count exceeds the number of direct access devices requested in the UNIT parameter, you should code PRIVATE, e.g., UNIT=(2311,4),VOLUME=(PRIVATE,,,6). When PRIVATE is coded and all the mounted volumes are used, the system demounts one of the volumes and then mounts another volume in its place so that processing can continue. When PRIVATE is not coded and all the mounted volumes are used, the system does not demount any of the volumes; therefore, the job step abnormally terminates. For tape devices, the PRIVATE subparameter is unnecessary; additional volumes are mounted as they are required.

The volume count subparameter is a positional subparameter. If you omit this subparameter, you code a comma to indicate its absence only if PRIVATE, RETAIN, or the volume sequence number subparameter is coded and the SER or REF subparameter follows.

SUPPLYING VOLUME SERIAL NUMBERS (SER)

To retrieve an existing data set, other than a cataloged or passed data set, you must supply the system with the serial numbers of the volumes on which the data set resides. When you are creating a data set, you can supply the system with the serial numbers of the volumes on which the data set will reside or let the system assign volumes to the data set. One of the ways to supply the system with serial numbers is to code the serial numbers on the DD statement. You can specify a maximum of 255 volume serial numbers per DD statement and a maximum of 4095 volume serial numbers per job step.

A volume serial number must be 1 to 6 characters in length. If volume serial number is not 6 characters, it will be padded with trailing blanks. It can contain any alphameric and national (#,$,@) characters, and the hyphen. You must enclose any volume serial number that includes special characters, other than a hyphen, in apostrophes whenever you code that number in the VOLUME parameter. When using

various typewriter heads or printer chains, difficulties in volume serial recognition may arise if you use other than alphameric characters. Each volume at an installation should have a different serial number regardless of the volume type, e.g., tape, disk; the volume's serial number should be posted on the outside of the volume.

The SER subparameter appears as the last subparameter in the VOLUME parameter. Follow SER= with the volume serial numbers. The serial numbers must be enclosed in parentheses, unless there is only one serial number. If SER is the only subparameter you are coding, you can code VOLUME=SER=(serial number,...) or VOLUME=SER=serial number.

SCRTCH should not be used as a volume serial number, because it is used to notify the operator to mount a non-specific volume. For Optical Readers, if no volume serial number is specified, VOLUME=SER=OCRINP is assumed.

REFERRING THE SYSTEM TO AN EARLIER SPECIFIC VOLUME REQUEST (REF)

Another way to supply the system with volume serial numbers is to refer the system to either a cataloged data set or a data set that is defined earlier in the job. When you do this, the system obtains volume information, including volume serial numbers, the label type field, and unit information from the source you refer it to. For Direct Access devices the label type is obtained from the label parameter specified in the DD statement and not from the source you refer it to. When using VOL=REF reference to a previous DD statement that uses esoteric names, the system will allocate to generic, rather than esoteric, names.

To refer the system to a cataloged data set or to a data set passed earlier in the job that has not been assigned a temporary data set name, you code REF as the last subparameter in the VOLUME parameter. Follow REF= with the data set name of the cataloged or passed data set. The data set name you code cannot contain special characters, except for periods used in a qualified name.

To refer the system to a data set defined earlier in the job that was not passed or was passed but assigned a temporary name, you code REF= as the last subparameter in the VOLUME parameter. Follow REF= with a backward reference to the DD statement that contains the volume serial numbers. This backward reference must be one of the following:

1. *.ddname. Use this form of backward reference when the DD statement you are referring to is contained in the same job step.

2. *.stepname.ddname. Use this form of backward reference when the DD statement you are referring to is contained in an earlier job step.

3. *.stepname.procstepname.ddname. Use this form of backward reference when the DD statement you are referring to is contained in a cataloged procedure step that is part of a procedure called by an earlier job step.

In any case, if the ddname refers to a DD statement that defines a dummy data set, the DD statement requesting use of the volumes assigned to that data set is assigned a dummy status.

When you refer the system to a data set that resides on more than one tape volume, the system assigns only the last volume. When you refer the system to a data set that resides on more than one direct access volume, the system assigns all of the volumes. In either case, you can code the volume count subparameter if additional volumes may be required.

If REF is the only subparameter you are coding, you can code VOLUME=REF=reference.

Volume Affinity

Two or more data sets sharing the same volume have volume affinity. This occurs when you specify the same volume serial numbers for the data sets, or when you use the REF subparameter of the VOLUME parameter to indicate that volumes identified in the catalog or on an earlier DD statement in the job are to be assigned to the data set being defined. Volume affinity influences device allocation. The system may ignore a request for a specific number of units if a data set has volume affinity with at least one other data set. The system will allocate devices as follows:

situation: The volume or volumes requested for a data set are shared.
The device requested is not a tape unit.
The data sets sharing volumes do not request unit affinity.

allocation: The system ignores any request for a specific number of units and allocates one device for each shared volume.

example: Two data sets share three volumes:

```
//UN      DD   UNIT=SYSDA,DSN=ANN,VOL=SER=(AA,BB,CC)
//DEUX    DD   UNIT=SYSDA,DSN=JOHN,VOL=SER=(AA,BB,CC)
```

Three devices are allocated to the data set defined by UN.

The same three devices are allocated to the data set defined by DEUX.

situation: Some of the volumes requested for a data set are shared.
The UNIT parameter requests more devices than the number of shared volumes.
The device requested is not a tape unit.
The data sets sharing volumes do not request unit affinity.

allocation: The system allocates the number of units requested in the UNIT parameter.

example: Two data sets share one volume:

```
//TROIS   DD   UNIT=(SYSDA,3),DSN=CYNDY,VOL=SER-(AA,BB,CC)
//QUATRE  DD   DSN=BERT,VOL=SER=BB
```

The UNIT parameter of TROIS requests 3 units; only one of the volumes requested by TROIS is shared. The system honors the unit request and allocates 3 advices to the data set defined by TROIS. The system allocated one device to the data set defined by QUATRE.

situation: Some of the volumes requested by a data set are shared.
The number of units requested for the data set is equal to or less than the number of shared volumes (or no request for a specific number of units is made).
The device requested is not a tape unit.
The DD statements sharing volumes do not request unit affinity.

allocation: The system will allocate one device for each shared volume plus one additional device.

example: Two data sets share three volumes:

```
//CINQ    DD   DSN=ALL,UNIT=(SYSDA,3),VOL=SER=(AA,BB,
               CC,DD,EE)
//SIX     DD   DSN=JANE,VOL=SER=(BB,DD,EE)
```

The system ignores the request for three devices made in the UNIT parameter of CINQ and allocates four devices -- one for each shared volume plus one additional device. Three devices are allocated to the data set defined by SIX.

situation: Two data sets share at least one volume.
The device requested is a tape unit.
One of the data sets requests unit affinity with the other data set.

allocation: The system will assign the number of units requested in the UNIT parameter. If no request for a specific number of units is made, the system assumes 1.

example: Two data sets share three volumes:

```
//SEPT    DD   UNIT=(2400,2),DSN=JAN,VOL=SER=(VOL1,
               VOL2,VOL3)
//HUIT    DD   UNIT=AFF=SEPT,DSN=MARTY,
               VOL=SER=(VOL1,VOL2,VOL3,VOL4)
```

The system allocates two devices to the data set defined by SEPT, and the same two devices to the data set defined by HUIT.

Note: If you code duplicate serial numbers in the VOLUME parameter, the system will treat that volume as a shared volume, and will allocate devices to the data set according to the above considerations.

Volume States

Every mounted volume is assigned several attributes by the system. The attributes assigned to a mounted volume define the state of the volume; the volume state controls when a volume is demounted and controls volume sharing. Volume sharing is the allocation of a volume to two or more data sets defined in the same job step, or the allocation of a direct access volume to two or more data sets defined in different job steps that are executing concurrently.

The attributes that are assigned both to a tape or direct access volume are the mount attribute and the use attribute. The nonsharable attribute can also be assigned to a direct access volume. These attributes are described in the next two topics.

THE MOUNT AND USE ATTRIBUTES

Every volume is assigned a mount and use attribute. The mount attribute controls volume demounting. The use attribute is one of the factors that controls allocation of mounted volumes to data sets. The mount and use attributes are:

Mount	Use
Permanently resident Reserved Removable	Public Private Storage Scratch

The following lists the mount attributes and describes how this attribute and a use attribute are assigned to a volume.

1. Permanently resident volumes cannot be demounted. Only direct access volumes can be permanently resident. While all direct access volumes can be designated as permanently resident in a special member of SYS1.PARMLIB named PRESRES, the following volumes are always permanently resident:

   - All volumes that cannot be physically demounted, such as a 2301 Drum Storage volume.
   - The volume from which the system is loaded (the IPL volume).
   - The volume containing the system data sets SYS1.LINKLIB, SYS1.PROCLIB, and SYS1.SYSJOBQE.

   A permanently resident volume can be assigned the use attribute of public, private, or storage. The use attribute is assigned to the volume in the PRESRES member in SYS1.PARMLIB, or is public by default.

2. Reserved volumes remain mounted until an UNLOAD command is issued. Both direct access and tape volumes can be reserved volumes. A volume becomes reserved as a result of a MOUNT command or a PRESRES entry. A volume is usually designated as a reserved volume to avoid repeated mounting and demounting of the volume when it is to be used by a group of related jobs.

   A reserved direct access volume can be assigned the use attribute of public, private, or storage. The use attribute is assigned to the volume either in the PRESRES member in SYS1.PARMLIB or in a parameter of the MOUNT command, depending on how the volume becomes reserved.

   A reserved tape volume is always assigned the use attribute of private.

3. Removable volumes are those volumes that are neither permanently resident nor reserved. Removable volumes are demounted either after their last use in a job step or when the unit on which the volume is mounted is required for another volume. Which occurs depends on the use attribute assigned to the volume.

   A removable direct access volume can be assigned the use attribute of public or private. The use attribute of public is assigned when the PRIVATE subparameter is not coded. The use attribute of private is assigned when the PRIVATE subparameter is coded.

   A removable tape volume can be assigned the use attribute of scratch or private. The use attribute of scratch is assigned when the PRIVATE subparameter is not coded, a nonspecific volume request is made, and the data set is temporary. The use attribute of private is assigned when the PRIVATE subparameter is coded, a specific volume request is made, or the data set is nontemporary.

   Note: If, when you make a nonspecific volume request for a tape with IBM standard labels, the system allocates a device containing a ready tape, the system will assume it is a scratch tape and use it. This tape could be available for the following reasons:

- The tape was left mounted as a scratch tape by another job because the disposition specified for the data set on that tape was DELETE.

- The tape had been requested by another job, but the job terminated before the tape became ready. As a result, no message to demount the tape was sent to the operator. This situation can be avoided by coding DEFER in the UNIT parameter to defer mounting of the volume until the processing program attempts to open the data set.

Figure 25 summarizes what type of volume can be assigned when you make a specific or nonspecific volume request for a temporary or nontemporary data set, how these attributes are assigned, and how the volume is demounted.

Volume State	Temporary Data Set: Type of Volume Request	Nontemporary Data Set: Type of Volume Request	How Assigned	How Demounted
Public/ Permanently Resident[1]	Nonspecific or Specific	Specific	PRESRES Entry or by default	Always mounted
Private/ Permanently Resident[1]	Specific	Specific	PRESRES Entry	Always mounted
Storage/ Permanently Resident[1]	Nonspecific or Specific	Nonspecific or Specific	PRESRES Entry	Always mounted
Public/ Reserved[1]	Nonspecific or Specific	Specific	PRESRES Entry or MOUNT command	UNLOAD command
Private/ Reserved (Tape and direct access)	Specific	Specific	PRESRES Entry or MOUNT command (Only MOUNT command for tape.)	UNLOAD command
Storage/ Reserved[1]	Nonspecific or Specific	Nonspecific or Specific	PRESRES Entry or MOUNT command	UNLOAD command
Public/ Removable[1]	Nonspecific or Specific	Specific	VOLUME=PRIVATE is not coded on the DD statement	When unit is required by another volume.
Private/ Removable (Tape and direct access)	Specific	Specific	VOLUME=PRIVATE is coded on the DD statement (Specific request or a nontemporary data set for tape also causes this assignment.)	After its use, unless RETAIN or PASS is coded, in which case, volume demounted at job termination
Scratch (Tape only)	Nonspecific or Specific	Nonspecific or Specific	Any tape data set (Scratch volume becomes private if VOLUME=PRIVATE is coded, specific request is made, or data set is nontemporary.)	When unit is required by another volume.

[1]Direct access volumes only.

Figure 25. Combinations of Mount and Use Attributes

NONSHARABLE ATTRIBUTE

The nonsharable attribute is assigned by the system to direct access volumes that may require demounting during execution of the step that requested the volume. When a volume is assigned the nonsharable attribute, the volume cannot be assigned to a data set defined in the same step for which a nonspecific request is made or to any data set defined in another step that is being executed concurrently.

The nonsharable attribute is never assigned to a permanently resident or reserved volume or to a volume that was mounted to satisfy a nonspecific request for a public volume. Except for these cases just described, the nonsharable attribute is always assigned to a volume when the following occurs:

1. You make a specific volume request and request more volumes than devices.

2. You request unit affinity with an earlier data set defined in the job step. (The data sets must reside upon different volumes.)

3. You request deferred mounting of the volume on which the data set resides.

4. You make a nonspecific request for a private volume.

SATISFYING SPECIFIC VOLUME REQUESTS

In the following cases the system can satisfy a request for a specific volume that is already mounted:

1. The volume is permanently resident or reserved. The use attribute of the volume does not affect assignment of the volume and the use attribute is not changed.

2. The direct access volume is a removable volume that has not been assigned the nonsharable attribute and is being used by a concurrently executing step. (If your request would make the volume nonsharable, the system waits to assign you that volume until all other job steps using the volume have terminated.) The volume remains private if its use attribute is private. The volume becomes private if the use attribute is public and the request is for a private volume. The volume remains public if its use attribute is public and the request is for a public volume.

3. The direct access volume is a removable public volume and is not in use. The use attribute (private or public) assigned to the volume when it is allocated is determined by the presence or absence of the PRIVATE subparameter.

4. The tape volume is a scratch volume and is not in use. The use attribute of private is assigned to the volume.

SATISFYING NONSPECIFIC VOLUME REQUESTS

There are four types of nonspecific volume requests that can be made:

1. You can request a private volume for a temporary data set.

2. You can request a private volume for a nontemporary data set.

3. You can request a public volume for a temporary data set.

4. You can request a storage volume for a nontemporary data set.

How the system satisfies these different types of requests are described below. Since the system satisfied the first two types of requests in the same way, these two requests are described together.

1. When you make a nonspecific volume request for a private direct access or tape volume, the system assigns a volume that is mounted but not in use or requests the operator to mount a volume. The operator should mount a volume whose space is unused. This allows you to have control over all space on the volume. Once mounted, the volume is assigned the use attribute of private.

2. When you make a nonspecific volume request for a public direct access volume that is to contain a temporary data set, the system assigns a public or storage volume that is already mounted, or requests the operator to mount a removable volume. If a mounted volume is selected, its use attribute is not affected. If a removable volume is mounted, it is assigned the use attribute of public.

   When you make a nonspecific volume request for a public tape volume that is to contain a temporary data set, the system assigns a scratch volume that is already mounted, or it requests the operator to mount a tape volume. Once mounted, the volume is assigned the use attribute of scratch.

3. When you make a nonspecific volume request for a public direct access volume that is to contain a nontemporary data set, the system assigns a storage volume if one is mounted. Otherwise, the request is treated as a nonspecific volume request for a private volume.

   When you make a nonspecific volume request for a public tape volume that is to contain a nontemporary data set, the request is treated as a nonspecific volume request for a private volume.

Examples of the VOLUME Parameter

1.
```
//DD1      DD   DSNAME=STEP,UNIT=2311,DISP=OLD,                    X
//             VOLUME=(PRIVATE,,,SER=548863)
```

   This DD statement defines an existing data set and informs the system that the data set resides on the volume whose serial number is 548863. Since PRIVATE is coded in the VOLUME parameter, the system will not assign the volume to any data set for which a nonspecific volume request is made and will cause the volume to be demounted after its use in the job step.

2.
```
//DDB      DD   DSNAME=COMM,DISP=(NEW,KEEP),SPACE=(CYL,(30,2)),    X
//             VOLUME=(PRIVATE,,,2),UNIT=2311
```

   The DD statement named DDB defines a new data set for which the system is to assign a volume. Since only one device is requested (UNIT=2311) and the volume count is 2, PRIVATE is coded to ensure that the additional volume can be mounted if required.

3.
```
//DD2      DD   DSNAME=QUET,DISP=(MOD,KEEP),UNIT=(2400,2),        X
//             VOLUME=(,,,4,SER=(96341,96342))
```

This DD statement defines an existing data set, which resides on the volumes whose serial numbers are 96341 and 96342, and requests that a total of 4 volumes be used to process the data set if required.

4.
```
//DD3      DD   DSNAME=&OUT,DISP=NEW,UNIT=2400
```

This DD statement defines a temporary data set and, by omission of the VOLUME parameter, requests the system to assign a suitable volume to the data set.

Device	Parameter Type	Parameter	Comments
Unit Record Devices	Location of the Data Set	UNIT	Required
	Data Attributes	DCB	Optional
	Special Processing Options	UCS	Optional (for a printer with the universal character set feature)
		FCB	Optional (for 3211 printer if forms control information is to be specified)
		DUMMY	Optional
System Output Devices	Location of the Data Set	SYSOUT	Required. Specifies the output class
		UNIT	Optional
	Size of the Data Set	SPACE	Optional
	Data Attributes	DCB	Optional
	Special Processing Option	OUTLIM	Optional. Meaningful only for Systems that have the Systems Management Facilities Option
Magnetic Tape	Data Information	DSNAME (or DSN)	Required if the data set is to be cataloged or used by a later job
		DISP	Required if the data set is to be cataloged, used by a later step in this job, or used by another job
	Location of the Data Set	UNIT	Required unless you request (with the VOLUME parameter) the same volume used for an earlier data set in your job
		VOLUME (or VOL)	Required if you want a specific volume. If you do not use this parameter you get a scratch tape
		LABEL	Required if you do not want to use IBM standard labels for the data set
	Data Attributes	DCB	Optional
	Special Processing Options	SEP	Either parameter can be used
		AFF	
		DUMMY	Optional
Direct Access Devices	Data Set Information	DSNAME (or DSN)	Required if the data set is to be cataloged or used by a later job
		DISP	Required if the data set is to be cataloged, used by a later step in this job, or used by another job
	Location of the Data Set	UNIT	Required unless you request (with the VOLUME parameter) the same volume used for an earlier data set in your job, or unless you use the SPLIT or SUBALLOC parameters to allocate space to this data set
		VOLUME (or VOL)	Required if you want a specific volume or multiple volumes. If you do not use this parameter your data set will be allocated on any suitable volume
		LABEL	Required if you want the data set to have both IBM standard and user labels
	Size of the Data Set	SPACE	One of these parameters is required. SPLIT can only be used for BSAM or QSAM data sets. SPACE must be used for ISAM data sets
		SPLIT	
		SUBALLOC	
	Data Attributes	DCB	Optional. Required for BDAM and ISAM data sets
	Special Processing Options	SEP	Either parameter can be used
		AFF	
		DUMMY	Optional
		DYNAM	Optional. Meaningful only for systems with MVT and TSO

Figure 26. Parameters for Creating A Data Set

SUMMARY

The following figures summarize the DD statement parameters required to perform these functions:

- Create a data set on an unit record device (card punch or printer) — Figure 27
- Create a data set on a system output device — Figure 28
- Create a data set on magnetic tape — Figure 29
- Create a data set on a direct access device — Figure 30
- Retrieve a data set from an unit record device (card reader or paper tape reader) — Figure 31
- Retrieve a data set from the input stream — Figure 32
- Retrieve a passed data set (magnetic tape or direct access) — Figure 33
- Retrieve a cataloged data set (magnetic tape or direct access) — Figure 34
- Retrieve a kept data set (magnetic tape or direct access) — Figure 35
- Extend a passed data set (magnetic tape or direct access) — Figure 36
- Extend a cataloged data set (magnetic tape or direct access) — Figure 37
- Extend a kept data set (magnetic tape or direct access) — Figure 38
- Postpone definition of a data set — Figure 39

Parameter Type	Parameter	Format
Location of the Data Set	UNIT	UNIT={unit address / 1052 / 1403 / 1442 / 1443 / 2520 / 2540-2 / 3211 / group name}
Data Attributes	DCB	[DCB=(list of attributes) DCB=({*.ddname / *.stepname.ddname / *.stepname.procstepname.ddname}[,list of attributes])] Note: See Figures 10 and 11 for attributes.
Special Processing Options	UCS	[UCS=(code [,FOLD][,VERIFY])]
	FCB	[FCB=(image-id [,ALIGN / ,VERIFY])]
	DUMMY	[DUMMY] Note: Positional parameter

Figure 27. Creating a Data Set on a Unit Record Device (Card Punch or Printer)

Parameter Type	Parameter	Format
Location of the Data Set	SYSOUT	SYSOUT=(classname [,program / ,][,form number]) Note: "Classname" is a letter (A-Z) or a number (0-9)
	UNIT	
Size of the Data Set	SPACE	
Data Attributes	DCB	
Special Processing Option	OUTLIM	OUTLIM=number Note: For systems with MFT or MVT that have the System Management Facilities option.

Figure 28. Creating a Data Set on a System Output Device

Parameter Type	Parameter	Format
Data Set Information	DSNAME and DISP	Temporary - for the duration of the job: DSNAME={&&name / &name},DISP=(NEW,PASS,DELETE) Temporary - for duration of the job step: DSNAME={&&name / &name},DISP=(NEW,DELETE,DELETE) Nontemporary - cataloged: DSNAME=dsname,DISP=(NEW,CATLG[,CATLG / ,DELETE / ,KEEP]) Nontemporary - kept: DSNAME=dsname,DISP=(NEW,KEEP[,KEEP / ,DELETE / ,CATLG]) Nontemporary - member generation data group DSNAME=groupname(+number),DISP=(NEW,CATLG[,CATLG / ,DELETE])
Location of the Data Set	UNIT	UNIT=(unit address / 2400 / 2400-1 / 2400-2 / 2400-3 / 2400-4 / group name / 3400)[,unitcount / ,P / ,][,DEFER])
	VOLUME	Nonspecific request - one scratch volume (temporary data set): Omit Nonspecific request - more than one scratch volumes (temporary data set): VOLUME=(,,,volcount) Nonspecific request - private volume (temporary data set): VOLUME=(PRIVATE[,,,volcount]) Nonspecific request - private volume (nontemporary data set): VOLUME=([PRIVATE,RETAIN / ,][,,volcount]) Specific request - private volume (request by serial number): VOLUME=([PRIVATE,RETAIN / ,][,,volcount, / ,]SER=(serial,...))

Figure 29. Creating a Data Set on a Magnetic Tape (Part 1 of 2)

Parameter Type	Parameter	Format
Location of the Date Set (cont.)	VOLUME (cont.)	Specific request - private volumes used by other data set): VOLUME=([PRIVATE,RETAIN][,,volcount,][,] REF={dsname / *.ddname / *.stepname.ddname / *.stepname.procstepname.ddname})
	LABEL	LABEL=([sequence] [,SL / ,SUL / ,NSL / ,NL / ,BLP] [,PASSWORD][,IN / ,OUT][,][EXPDT=yyddd / RETPD=nnnn])
Data Attributes	DCB	[DCB=(list of attributes) DCB=({*.ddname / *.stepname.ddname / *.stepname.procstepname.ddname}[,list of attributes])] Note: See Figure 12 for attributes.
Special Processing Options	SEP or AFF	[SEP=(ddname,...) / AFF=ddname]
	DUMMY	[DUMMY] Note: Positional parameter

Figure 29. Creating a Data Set on Magnetic Tape (Part 2 of 2)

Parameter Type	Parameter	Format
Data Set Information	DSNAME and DISP	Temporary - for the duration of the job: DSNAME={&&name / &name / &&name(membername) / &name(membername)},DISP=(NEW,PASS,DELETE)) Temporary - for the duration of the job step: DSNAME={&&name / &name / &&name(membername) / &name(membername)},DISP=(NEW,DELETE,DELETE) Nontemporary - cataloged: DSNAME={dsname / dsname(membername) / dsname(membername)},DISP=(NEW,CATLG[,CATLG / ,DELETE / ,KEEP]) Nontemporary - kept: DSNAME={dsname / dsname(membername)},DISP=(NEW,KEEP[,KEEP / ,DELETE / ,CATLG]) Nontemporary - member generation data group: DSNAME=groupname(+number),DISP=(NEW,CATLG[,CATLG / ,DELETE])
Location of the Data Set	UNIT	UNIT=({unit address / 2301 / 2302 / 2303 / 2305-1 / 2305-2 / 2311 / 2314 / 2319 / 2321 / 3330 / group name}[,unitcount / ,P / ,][,SEP=(ddname,...)]) Note: To indicate the 2319, specify UNIT=2314. To designate the 2319 as the particular device for your data set, specify UNIT=unit address.
	VOLUME	Nonspecific request - one public volume: Omit Nonspecific request - more than one public volumes: VOLUME=(,,,volcount) Nonspecific request - private volume: VOLUME=(PRIVATE[,RETAIN / ,][,,volcount]) Specific request - public volume (request by serial number): VOLUME=([,,,volcount,]SER=(serial,...))

Figure 30. Creating a Data Set on Direct Access Devices (Part 1 of 3)

Parameter Type	Parameter	Format
Location of the Data Set (cont.)	VOLUME (cont.)	Specific request - public volume (volumes used by other data set): VOLUME=([,,,volcount,]REF={dsname / *.ddname / *.stepname.ddname / *.stepname.procstepname.ddname}) Specific request - private volume (request by serial number): VOLUME=(PRIVATE[,RETAIN][,,volcount,]SER=(serial,...)) Specific request - private volume (volumes used by other data set): VOLUME=(PRIVATE[,RETAIN][,,volcount,] REF={dsname / *.ddname / *.stepname.ddname / *.stepname.procstepname.ddname})
	LABEL	[LABEL=([,SL / ,SUL][,PASSWORD][,IN / ,OUT][,][EXPDT=yyddd / RETPD=nnnn])]
Size of the Data Set (choose one)	SPACE	The system assigns tracks: SPACE=({TRK / CYL / block length},(primary quantity[,secondary quantity] [,directory])[,RLSE][,CONTIG / ,MXIG / ,ALX][,ROUND]) Requesting specific tracks: SPACE=(ABSTR,(primary quantity,address[,directory]))
	SPLIT	Units of cylinders: • First data set SPLIT=(n,CYL,(primary quantity [,secondary quantity])) • Subsequent data sets SPLIT=n Units of blocks: • First data set SPLIT=(%,blocklength,(primary quantity[,secondary quantity])) • Subsequent data sets SPLIT=%

Figure 30. Creating a Data Set on Direct Access Devices (Part 2 of 3)

Parameter Type	Parameter	Format
Size of the Data Set choose one) (cont.)	SUBALLOC	Reserved area: SPACE=({TRK / CYL / blocklength},(primary quantity),,CONTIG) Suballocaed data sets: SUBALLOC={TRK / CYL / block length},(primary quantity [,secondary quantity] [,directory]) {ddname / ,stepname.ddname / stepname.procstepname.ddname})
Data Attributes	DCB	[DCB=(list of attributes) DCB=({dsname / *.ddname / *.stepname.ddname / *stepname.procstepname.ddname} [,list of attributes])] Note: See Figure 13 for attributes.
Special Processing Options	SEP or AFF	[SEP=(ddname,...) AFF=ddname]
	DUMMY	[DUMMY] Note: Positional parameter
	DYNAM	[DYNAM] Note: Positional parameter

Figure 30. Creating a Data Set on Direct Access Devices (Part 3 of 3)

Parameter Type	Parameter	Format
Location of the Data Set	UNIT	UNIT={unit address / 1442 / 2520 / 2540 / 2671 / group name}
Data Attributes	DCB	[DCB=(list of attributes)] [DCB=({*.ddname / *.ddname.stepname / *.ddname.procstepname.ddname}[,list of attributes])]
Special Processing Option	DUMMY	[DUMMY] Note: Positional parameter

Figure 31. Retrieving an Existing Data Set From a Unit Record Device (Card Reader or Paper Tape Reader)

Parameter Type	Parameter	Format
Location of the Data Set	* or DATA	{* / DATA}
Data Attributes	DCB	[DCB=([BLKSIZE=number of bytes][,][BUFNO=number of buffers])]
Special Processing Option	DLM	DLM=delimiter

Figure 32. Retrieving A Data Set From the Input Stream

Parameter Type	Parameter	Format
Data Set Information	DSNAME	Retrieving by name: DSNAME={dsname | dsname(membername) | &&name | &&name(membername) | &name | &name(membername)} Making backward reference: DSNAME={*.ddname | *.stepname.ddname | *.stepname.procstepname.ddname}
	DISP	Passing the data set to a subsequent step: DISP=(OLD,PASS[,DELETE | ,CATLG | ,KEEP]) Last time a passed temporary data set is used in job: DISP=(OLD,DELETE,DELETE) Last time a passed nontemporary data set is used in job: DISP=(OLD[,KEEP | ,DELETE | ,CATLG | ,][,KEEP | ,DELETE | ,CATLG])
Location of the Data Set	UNIT	[UNIT=(,unitcount)]
	LABEL	[LABEL={,SUL | ,NL | ,NSL | ,BLP}]
Data Attributes	DCB	[DCB=(list of attributes) DCB={*.ddname | *.stepname.ddname | *.stepname.procstepname.ddname}] Note: See Figures 19, 20, 21, and 22 for attributes.
Special Processing Option	DUMMY	[DUMMY] Note: Positional parameter

Figure 33. Retrieving a Passed Data Set (Magnetic Tape or Direct Access)

Parameter Type	Parameter	Format
Data Set Information	DSNAME	DSNAME={dsname / dsname(membername) / groupname(number)}
	DISP	DISP=({OLD / SHR}[,UNCATLG / ,DELETE / ,][,UNCATLG / ,DELETE])
Location of the Data Set	UNIT	Requesting units: UNIT=(,[unitcount / P][,DEFER]) Unit affinity: [UNIT=AFF=ddname]
	VOLUME	VOLUME=([PRIVATE[,RETAIN / ,][,sequence])
	LABEL	[LABEL=(,{SUL / NL / NSL / BLP / LTM})]
Data Attributes	DCB	[DCB=(list of attributes) DCB=({dsname / *.ddname / *.stepname,ddname / *.stepname.procstepname.ddname}[,list of attributes])] Note: See Figures 19, 20, 21, and 22 for attributes.
Special Processing Options	SEP or AFF	[SEP=(ddname,...) / AFF=ddname]
	DUMMY	[DUMMY] Note: Positional parameter

Figure 34. Retrieving a Cataloged Data Set (Magnatic Tape or Direct Access)

Parameter Type	Parameter	Format
Data Set Information	DSNAME	DSNAME={dsname / dsname(membername)}
	DISP	DISP=({OLD / SHR}[,KEEP / ,CATLG / ,DELETE / ,][,KEEP / ,CATLG / ,DELETE])
Location of the Data Set	UNIT	Requesting units: UNIT=({unit address / device type / group name}[,unitcount / ,P / ,][,DEFER]) Unit affinity: UNIT=AFF=ddname
	VOLUME	VOLUME=([PRIVATE][,RETAIN / ,]SER=(serial,...))
	LABEL	[LABEL=([sequence#][,SUL / ,NL / ,NSL / ,BLP / ,LTM])]
Data Attributes	DCB	[DCB=(list of attributes) DCB=({dsname / *.ddname / *.stepname.ddname / *.stepname.procstepname.ddname}[,list of attributes])] Note: See Figures 19, 20, 21, and 22 for attributes.
Special Processing Options	SEP or AFF	[SEP=(ddname,...) AFF=ddname]
	DUMMY	[DUMMY] Note: Positional parameter

Figure 35. Retrieving a Kept Data Set (Magnetic Tape or Direct Access)

Parameter Type	Parameter	Format
Data Set Information	DSNAME	Retrieving by name: DSNAME={dsname / dsname(membername) / &&name / &&name(membername) / &name / &name(membername)} Making a backward reference: DSNAME={*.ddname / *.stepname.ddname / *.stepname.procstepname.ddname}
	DISP	Passing the data set to a subsequent step: DISP=(MOD,PASS[,DELETE / ,CATALG / ,KEEP]) last time a passed temporary data set is used in job: DISP=(MOD,DELETE,DELETE) last time a passed nontemporary data set is used in job: DISP=(MOD[,KEEP / ,DELETE / ,CATLG / ,][,KEEP / ,DELETE / ,CATLG])
Location of the Data Set	UNIT	[UNIT=([,unitcount / ,][,DEFER])]
	VOLUME	[VOLUME=(,,,volcount)]
	LABEL	[LABEL=(,SUL / ,NL / ,NSL / ,BLP / ,LTM)]
Size of the Data Set	SPACE	[SPACE=({TRK / CYL / block length},(1,secondary quantity),RLSE)]
Data Attributes	DCB	[DCB=(list of attributes)] [DCB={*.ddname / *.stepname.ddname / *.stepname.procstepname.ddname}[,list of attributes]]
Special Processing Option	DUMMY	[DUMMY] Note: Positional parameter

Figure 36. Extending a Passed Data Set (Magnetic Tape or Direct Access)

Parameter Type	Parameter	Format
Data Set Information	DSNAME	DSNAME={dsname / dsname(membername)}
	DISP	DISP=(MOD[,UNCATLG / ,DELETE / ,CATLG / ,][,UNCATLG / ,DELETE / ,CATLG])
Location of the Data Set	UNIT	Requesting units: [UNIT=(,[unitcount / P][,DEFER])] Unit affinity: [UNIT=AFF=ddname]
	VOLUME	VOLUME=([PRIVATE[,RETAIN / ,][,sequence# / ,][,volcount])
	LABEL	[LABEL=(,SUL / ,NL / ,NSL / ,BLP)]
Size of the Data Set	SPACE	[SPACE=({TRK / CYL / block length},(1,secondary quantity),RLSE)]
Data Attributes	DCB	[DCB=(list of attributes) DCB=({dsname / *.ddname / *.stepname.ddname / *.stepname.procstepname.ddname}[,list of attributes])]
Special Processing Options	SEP or AFF	[SEP=(ddname,...) AFF=ddname]
	DUMMY	[DUMMY] Note: Positional parameter

Figure 37. Extending a Cataloged Data Set (Magnetic Tape or Direct Access)

Parameter Type	Parameter	Format
Data Set Information	DSNAME	DSNAME={dsname / dsname(membername)}
	DISP	DISP=(MOD[,KEEP / ,CATLG / ,DELETE / ,][,KEEP / ,CATLG / ,DELETE])
Location of the Data Set	UNIT	Requesting units: UNIT=({unit address / device type / group name}[,unitcount / ,P / ,][,DEFER]) Unit affinity: UNIT=AFF=ddname
	VOLUME	VOLUME=(PRIVATE[,RETAIN / ,][,volcount,]SER=(serial,...))
	LABEL	[LABEL=([sequence#][,SUL / ,NL / ,NSL / ,BLP / ,LTM])]
Size of the Data Set	SPACE	[SPACE=({TRK / CYL / block length}(1,secondary quantity),RLSE)]
Data Attributes	DCB	[DCB=(list of attributes) DCB=({dsname / *.ddname / *.stepname.ddname / *.stepname.procstepname.ddname}[,list of attributes])]
Special Processing Options	SEP or AFF	[SEP=(ddname,...) AFF=ddname]
	DUMMY	[DUMMY] Note: Positional parameter

Figure 38. Extending a Kept Data Set (Magnetic Tape or Direct Access)

Parameter Type	Parameter	Format
Special Processing Option	DDNAME	DDNAME=ddname
Data Attributes	DCB	[DCB=([BLKSIZE=number of bytes][,][BUFNO=number of buffers)]

Figure 39. Postponing Definition of a Data Set

Section V: The COMMAND Statement

Commands are issued to communicate with and control the system. All commands may be issued to the system via the operator's console; some commands may be also issued via a command statement in the input stream. In most cases, the operator issues the command. If you include a command statement as part of your job control statements, the command is usually executed as soon as it is read. (Disposition of commands read from an input stream is specified as a PARM parameter field in the cataloged procedure for the input reader.) Since a command is usually executed as soon as it is read, it is not likely that the command will be synchronized with the execution of the job step to which it pertains. Therefore, you should tell the operator which commands you want issued and when they should be issued, and let him issue them.

A command statement may appear immediately before a JOB statement, an EXEC statement, a null statement, or another command statement.

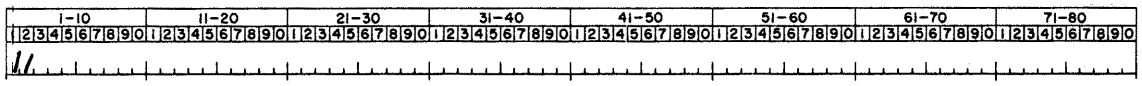
The Command Statement Format

```
//  command  operand  comments
```

The command statement consists of the characters // in columns 1 and 2, and three fields -- the operation (command), operand, and comments fields.

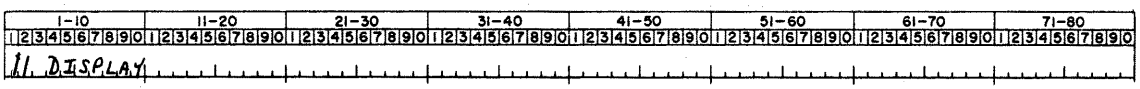
Rules for Coding

Code the command statement in the following order:



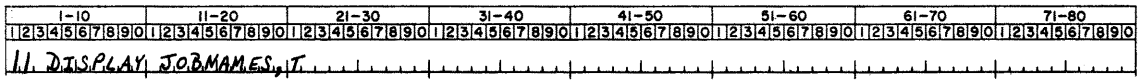
2. Follow // with one or more blanks.

3. Code the command.



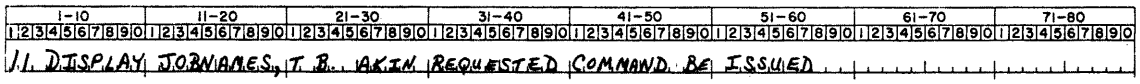
4. Follow the command with one or more blanks.

5. Code any required operands following the blank or blanks. Separate each operand with a comma.



6. Follow the operands with one or more blanks.

7. Code any comments following the blank or blanks.



8. The command statement cannot be continued.

Commands That Can Be Entered Through the Input Stream

The commands that can be entered through the input stream in MFT or MVT are listed below, with a brief explanation of what each command requests the system to do. Most command statements consist of an operation (command) field and an operand field, which includes options associated with the command. The operand field is not described here; a complete discussion of the commands and operands is presented in the Operator's Guide publication.

MFT

In MFT, the following commands can be entered through the input stream.

CANCEL: The CANCEL command tells the system to immediately terminate the scheduling or execution of a job, to cancel a job on the queue, or to stop the writing of an output data set currently being processed by an output writer.

DISPLAY: The DISPLAY command causes a console display of certain system status information.

HOLD: The HOLD command causes the system to temporarily prevent one job or all jobs from being selected for processing.

LOG: The LOG command is used to enter information into the system log.

MODIFY: The MODIFY command tells the system to change the characteristics of a functioning output writer.

MOUNT: The MOUNT command tells the system to assign a device so a particular volume can be mounted on it. This device can then be assigned by the system to any job step that requires that volume.

RELEASE: The RELEASE command tells the system to resume job selection, which had been suspended by the HOLD command or TYPRUN=HOLD on the JOB statement.

REPLY: The REPLY command is used to reply to messages from the system or from a processing program that requests information.

RESET: The RESET command tells the system to change the class or priority, or both, of a job in an input, hold, or system output queue.

SET: The SET command is used to establish the values of certain variables, such as the time of day and the date.

START: The START command tells the system to start a particular system process, e.g., an input reader, graphic job processor, initiator, etc.

STOP: The STOP command tells the system to stop a system process that had been previously started by a START command, or to stop the console display effected by the DISPLAY command.

UNLOAD: The UNLOAD command tells the system to remove the volume previously mounted in response to a MOUNT command.

VARY: The VARY command tells the system to place an I/O device or path into an online or offline status.

WRITELOG: The WRITELOG command tells the system to have the system output writer write out the contents of the system log.

MVT

In MVT, the following commands can be entered through the input stream.

CANCEL: The CANCEL command tells the system to immediately terminate the scheduling or execution of a job, to cancel a job on the queue, or to stop the writing of an output data set currently being processed by an output writer.

DISPLAY: The DISPLAY command causes a console display of certain system status information.

HOLD: The HOLD command causes the system to temporarily prevent one job or all jobs from being selected for processing.

LOG: The LOG command is used to enter information into the system log.

MODIFY: The MODIFY command tells the system to change the characteristics of a functioning initiator or output writer.

MOUNT: The MOUNT command tells the system to assign a device so a particular volume can be mounted on it. This device can then be assigned by the system to any job step that requires that volume.

RELEASE: The RELEASE command tells the system to resume job selection, which had been suspended by the HOLD command or TYPRUN=HOLD on the JOB statement.

REPLY: The REPLY command is used to reply to messages from the system or from a processing program that requests information.

RESET: The RESET command tells the system to change the class or priority, or both, of a job in an input, hold, or system output queue.

SET: The SET command is used to establish the values of certain variables, such as the time of day and the date.

START: The START command tells the system to start a particular system process, e.g., an input reader, graphic job processor, initiator, etc.

STOP: The STOP command tells the system to stop a system process that had been previously started by a START command or to stop the console display effected by the DISPLAY command.

UNLOAD: The UNLOAD command tells the system to remove the volume previously mounted in response to a MOUNT command.

VARY: The VARY command tells the system to place an I/O device or path into an online or offline status. In a Model 65 multiprocessing system (M65MP), this command is used to place I/O devices, paths, CPU, channel, and storage units in online or offline status.

WRITELOG: The WRITELOG command tells the system to have the system output writer write out the contents of the system log.

Example of the Command Statement

```
1.  //   START INIT,,,AB START AN INITIATOR FOR MFT
```

This command tells the system to start an initiator. The characters A and B indicate that the initiator is to select for execution only jobs of job classes A and B.

Section VI: The COMMENT Statement

The comment statement can be used to contain information that may be helpful to yourself or another person that may be running your job or reviewing your output listing.

The comment statement may appear anywhere except before the JOB statement or between continuation cards composing a single but extended JCL statement. A comment statement cannot be continued using continuation conventions; however, it can be followed by one or more comment statements.

The Comment Statement Format

```
//*comments
```

The comment statement consists of the characters //* in columns 1, 2, and 3, and the comments field.

Rules for Coding

Code the comment statement in the following order:

1. Code //* in columns 1, 2, and 3.

```
1-10      11-20     21-30     31-40     41-50     51-60     61-70     71-80
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
//*
```

2. Code the comments in columns 4 through 80.

```
1-10      11-20     21-30     31-40     41-50     51-60     61-70     71-80
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
//*THIS DATA SET WILL BE USED BY PROJECT 18 WHILE TESTING NEW PROGRAM TPSECRET.
```

3. If all of the comments cannot be included on this comment statement, follow it with another comment statement.

OUTPUT LISTINGS

In the MSGLEVEL parameter, you can request an output listing of all the control statements processed in your job. If you do, you can identify comment statements by the appearance of *** in columns 1, 2, and 3.

Example of the Comment Statement

1.
```
//*THE COMMENT STATEMENT CANNOT BE CONTINUED,
//*BUT IF YOU HAVE A LOT TO SAY, YOU CAN FOLLOW A
//*COMMENT STATEMENT WITH ONE OR MORE COMMENT
//*STATEMENTS.
```

Section VII: The DELIMITER Statement

When you submit data through an input stream, you must indicate to the system the beginning of the data and the end of the data. The beginning of the data is indicated by a DD * or DD DATA statement. The end of the data is indicated by a delimiter statement. The delimiter statement, however, is not required if the data is preceded by a DD * statement and you do not code the DLM parameter.

The Delimiter Statement Format

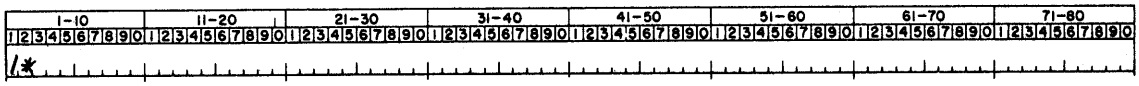
```
/* comments
```

The delimiter statement consists of the characters /* in columns 1 and 2 and the comments field. The system will recognize a delimiter other than /* if you code the DLM parameter on the DD statement defining data in the input stream. For more information, see the description of the DLM parameter in the section describing the DD statement.

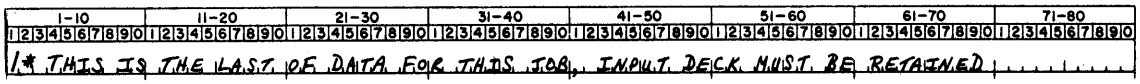
Rules for Coding

Code the delimiter statement in the following order:

1. Code /* (or the value assigned in the DLM parameter) in columns 1 and 2.



2. Code any desired comments.



3. The comments cannot be continued.

Example of the Delimiter Statement

1.
```
//JOB54   JOB,'C BROWN',MSGLEVEL=(2,0)
//STEPA   EXEC PGM=SERS
//DD1     DD  *
          .
          .
          .
          data
          .
          .
          .
/* END OF DATA FOR THIS STEP
```

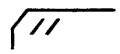
Section VIII: The NULL Statement

The null statement can be placed at the end of a job's control statements and data or at the end of all the statements in an input stream. The null statement tells the system that the job just read should be placed on the queue of jobs ready for processing. If there are any control statements or data between a null statement and the next JOB statement, these are flushed by the system.

If you do not follow your job's control statements and data with a null statement, the system places your job on the queue when it encounters another JOB statement in the input stream. If your job is the last job in the input stream and a null statement does not follow it, the system recognizes that this is the last job in the input stream and it places your job on the queue.

If a null statement follows a control statement that is being continued, the system treats the null statement as a blank comment field and assumes that the control statement contains no other operands.

The Null Statement Format



The null statement consists only of the characters // in columns 1 and 2. The remainder of the statement must be blank.

Example of the Null Statement

```
1.  //MYJB    JOB  ,'C DAVIS',MSGLEVEL=(1,1)
    //STEP1   EXEC PROC=FIELD
    //STEP2   EXEC PGM=XTRA
    //DD1     DD   UNIT=2400
    //DD2     DD   *
                 .
                 .
                 .
          data
                 .
                 .
                 .
    /*
    //
```

Section IX: The PEND Statement

The PEND statement is used to mark the end of an in-stream procedure. The name field of the PEND statement can contain a name. If comments are to be used, a blank must separate the operation field from the comment field. The PEND statement may not be continued. Do not include the PEND statement in cataloged procedures.

The PEND Statement Format

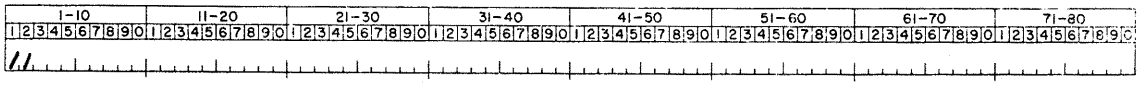
```
//name PEND comments
```

The PEND statement consists of the characters // in column 1 and 2 and three fields -- the name field, the operation (PEND) field, and the comments field.

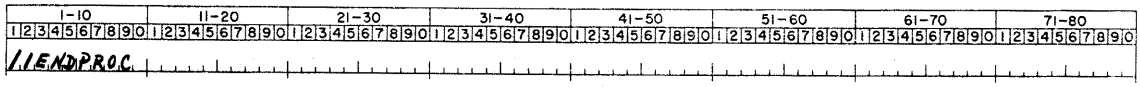
Rules for Coding

Code the PEND statement in the following order:

1. Code // in columns 1 and 2.

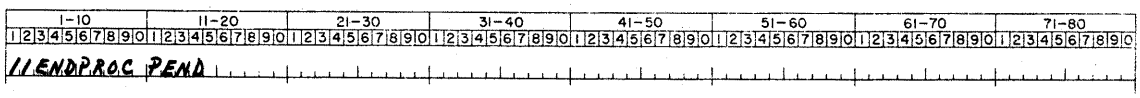


2. Follow // with a 1- to 8-character name or one or more blanks.



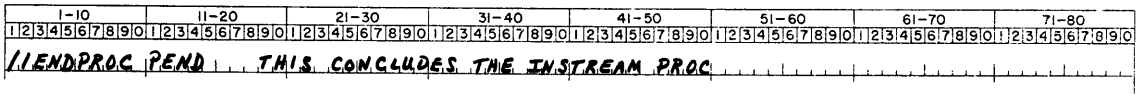
3. If a name is coded,
   Follow the name with one or more blanks.

4. Code PEND.



5. Follow PEND with one or more blanks.

6. Code any desired comments following the blank or blanks.



A PEND statement cannot be continued.

Examples of the PEND Statement

1. `//PROCEND1 PEND THIS STATEMENT IS REQUIRED FOR INSTREAM`

This PEND statement contains a comment.

2. `// PEND`

A PEND statement can contain only the coded operation field preceded by // and one or more blanks and followed by blanks.

Section X: The PROC Statement

The PROC statement is the first control statement in an in-stream procedure. Optionally, the PROC statement can also be the first control statement in a cataloged procedure. If a PROC statement is included in a cataloged procedure, it is used to assign default values for symbolic parameters in the procedure. In an in-stream procedure, the PROC statement is used to mark the beginning of the procedure and can be used to assign default values to symbolic parameters in the procedure. A default value appearing on a PROC statement can be overridden by assigning a value to the same symbolic parameter on the EXEC statement that calls the procedure.

The PROC Statement Format

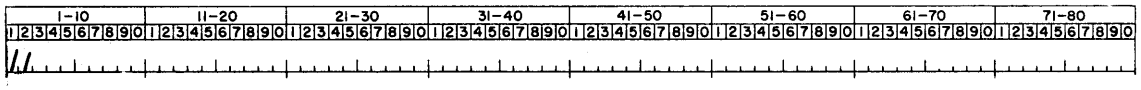
```
//name PROC operands comments
```

The PROC statement consists of the characters // in columns 1 and 2 and four fields -- the name field, the operation (PROC) field, the operand field, and the comments field.

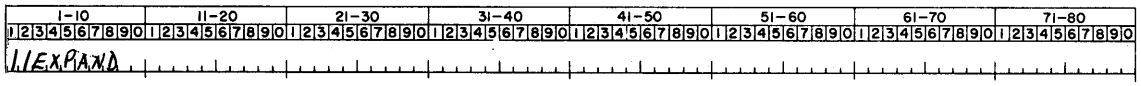
Rules for Coding

Code the PROC statement in the following order:

1. Code // in columns 1 and 2.

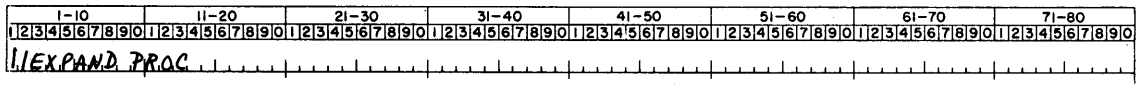


2. Follow // with a 1- to 8-character name or one or more blanks. A name is required for in-stream procedures.



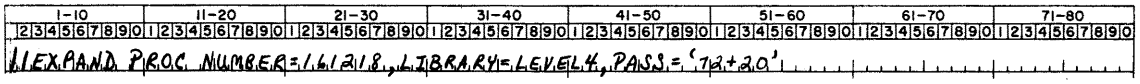
3. If a name is coded, follow the name with one or more blanks.

4. Code PROC.



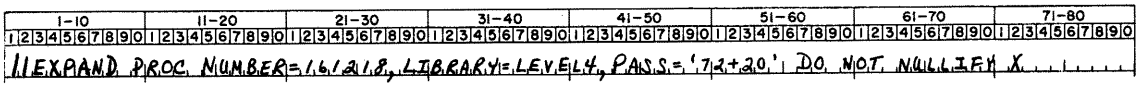
5. Follow PROC with one or more blanks.

6. Code the symbolic parameters and their default values following the blank or blanks. Separate each symbolic parameter and its default value with a comma. In a cataloged procedure, this field is not optional. In an in-stream procedure, this field is optional; if no operands are included, comments may not be coded, unless they appear on a continuation card.

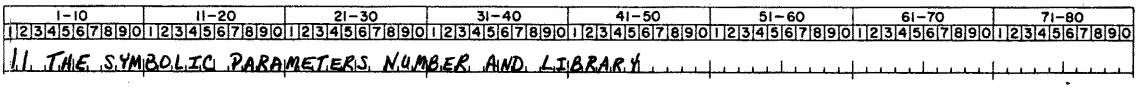


7. Follow the operands with one or more blanks.

8. Code any desired comments following the blank or blanks.



9. The PROC statement can be continued onto another statement.



If PROC statement is to be included in a cataloged procedure, it must appear as the first control statement. For an in-stream procedure, the PROC statement is required; it must appear as the first control statement of the in-stream procedure.

Assigning a Value on a PROC Statement to a Symbolic Parameter

To assign a value on a PROC statement to a symbolic parameter, code:

```
symbolic parameter=value
```

Omit the ampersand that precedes the symbolic parameter in the procedure.

You can also nullify a symbolic parameter on the PROC statement. Code:

```
symbolic parameter=
```

Omit the ampersand that precedes the symbolic parameter and do not follow the equal sign with a value.

There are some things you should keep in mind as you assign values to symbolic parameters:

1. The value you assign can be any length, but it cannot be continued onto another statement.

2. If the value contains special characters, enclose the value in appostrophes (the enclosing apostrophes are not considered part of the value). If the special characters include apostrophes, each must be shown as two consecutive apostrophes.

3. If you assign more than one value to a symbolic parameter on the PROC statement, the first value encountered is assigned.

4. If the symbolic parameter is concatenated with some other information (e.g., &JOBNO.321), this information and the value you assign to the symbolic parameter cannot exceed a combined total of 120 characters.

Example of the PROC Statement

1.
```
//DEF      PROC STATUS=OLD,LIBRARY=SYSLIB,NUMBER=777777
//NOTIFY   EXEC PGM=ACCUM
//DD1      DD   DSNAME=MGMT,DISP=(&STATUS,KEEP),UNIT=2400,        X
//             VOLUME=SER=888888
//DD2      DD   DSNAME=&LIBRARY,DISP=(OLD,KEEP),UNIT=2311,        X
//             VOLUME=SER=&NUMBER
```

Three symbolic parameters are defined in this cataloged procedure: &STATUS, &LIBRARY, and &NUMBER. Values are assigned to the symbolic parameters on the PROC statement. These values are used when the procedure is called and values are not assigned to the symbolic parameters by the programmer.

2.
```
//CARDS    PROC
```

This PROC statement can be used to mark the beginning of an in-stream procedure named CARDS.

Section XI: Appendixes

Appendix A: Cataloged and In-Stream Procedures

A cataloged procedure is a set of job control statements that has been assigned a name and placed in a partitioned data set known as the procedure library. (The IBM-supplied procedure library is named SYS1.PROCLIB; at your installation, there may be additional procedure libraries, which would have different names.) An in-stream procedure is a set of job control statements in the form of card images that have been placed in the input stream. An in-stream procedure can be executed any number of times during the job in which it appears. Both cataloged and in-stream procedures can consist of one or more steps; each step is called a procedure step. Each procedure step consists of an EXEC statement and DD statements. The EXEC statement identifies to the system what program is to be executed. The DD statements define the data sets to be used by the program.

You can use a cataloged procedure by coding the procedure name on an EXEC statement. You can use an in-stream procedure by coding the procedure name that is on the PROC statement on an EXEC statement. With both cataloged and in-stream procedures, you can follow this EXEC statement with DD statements that modify the procedure for the duration of the job step.

Appendix A consists of two chapters. The first chapter "Using Cataloged and In-stream Procedures" describes how to call a procedure, how to assign values to symbolic parameters, how to override parameters on the EXEC and DD statement, and how to add DD statements to a procedure. The second chapter "Writing Procedures: Cataloged and In-stream" describes the makeup of a procedure, how to use symbolic parameters, how to place a set of job control statements in the procedure library, and how to modify a procedure.

Using Cataloged and In-Stream Procedures

How to Call a Cataloged Procedure

To use a cataloged procedure, submit a JOB statement followed by an EXEC statement. On the EXEC statement you identify the cataloged procedure in one of two ways:

1. Code, as the first operand, the name assigned to the procedure; or

2. Code PROC= followed by the name assigned to the procedure as the first operand.

When you call a procedure, the system finds the control statements in the procedure library and then executes the programs identified on the EXEC statements in the procedure.

Besides identifying the procedure on the EXEC statement, you can assign values to symbolic parameters and override parameters that are coded on the EXEC statements contained in the procedure. You follow the EXEC statement with DD statements when you want to override DD statements in the procedure or add DD statements to the procedure.

When a cataloged procedure is written as part of the system output listing (i.e., MSGLEVEL=(1,0), MSGLEVEL=(1,1), or MSGLEVEL=1 is coded on the JOB statement), the procedure statements can be easily identified. An XX appears in columns 1 and 2 of a procedure statement that you did not override; X/ appears in columns 1 and 2 of a procedure statement that you did override; XX* appears in columns 1 and 2 of a procedure statement, other than a comment statement, that the system considered to contain only comments; and *** appears in columns 1 through 3 of a comment statement. In addition, if the procedure contains symbolic parameters, the output listing will show the symbolic parameters and the values assigned to them.

How to Call an In-Stream Procedure

To use an in-stream procedure, include the procedure, beginning with a PROC statement and ending with a PEND statement, with the job control language for your job. The in-stream procedure can appear immediately following the JOB statement, the JOBLIB DD statement, or the SYSCHK DD statement. The in-stream procedure cannot appear before the JOB statement or after the EXEC statement that calls it. An in-stream procedure can appear after a SYSIN DD * statement; however, this is not advisable because the SYSIN DD * statement causes the input reader to obtain direct access space for a system input data set.

To call the procedure, you identify the in-stream procedure on an EXEC statement in one of two ways:

1. Code, as the first operand, the name on the PROC statement of the procedure; or

2. Code PROC= followed by the name on the PROC statement of the procedure.

When you call an in-stream procedure, the system finds the control statements that have been written on a direct access device and then executes the programs identified on the EXEC statements of the procedure.

Besides identifying the procedure on the EXEC statement, you can assign values to symbolic parameters and override parameters that are coded on the EXEC statements contained in the procedure. You follow the EXEC statement with DD statements when you want to override DD statements in the procedure or add DD statements to the procedure.

When an in-stream procedure is written as part of the system output listing (i.e., MSGLEVEL=(1,0), MSGLEVEL=(1,1), MSGLEVEL=1, or MSGLEVEL=2 is coded on the JOB statement), the procedure statements can be easily identified. An ++ appears in columns 1 and 2 of a procedure statement that you did not override; +/ appears in columns 1 and 2 of a procedure statement that you did override; +++ appears in column 1 through 3 of a procedure statement, other than a comment statement, that the system considered to contain only comments; and *** appears in columns 1 through 3 of a comment statement. In addition, if the procedure contains symbolic parameters and you assign values to these on the EXEC statement that calls the procedure, the output listing will show the symbolic parameters and the values assigned to them.

Assigning Values to Symbolic Parameters

The cataloged or in-stream procedure you call may contain symbolic parameters. A symbolic parameter is characterized by a name preceded by an ampersand (&) and appears in the operand field of a cataloged or in-stream procedure statement or a DD statement used to override a DD statement in the procedure. A symbolic parameter stands as a symbol for a parameter, a subparameter, or a value. Symbolic parameters are used so that the procedure can be modified easily when it is called by a job step.

The following are examples of symbolic parameters:

```
//STEP1 EXEC PGM=COB,PARM='P1,&P2,,P3'

//DD1 DD DSNAME=FIX,UNIT=&DEVICE,SPACE=(CYL,(&SPACE,10))

//DD2 DD DSNAME=CHAG,UNIT=2400,DCB=BLKSIZE=&LENGTH
```

Symbolic parameters must either be assigned values or nullified before the procedure is executed. There are two ways that a symbolic parameter can be assigned a value:

1. A value can be assigned to the symbolic parameter on the EXEC statement that calls the procedure.

2. The PROC statement, which can appear as the first statement in a cataloged procedure and must appear as the first statement in an in-stream procedure, assigns a default value to the symbolic parameter.

Any default value assigned to a symbolic parameter on the PROC statement is overridden when you assign a value to the same symbolic parameter on the EXEC statement that calls the procedure. Symbolic parameters within quotes in the PARM field are an exception. Normally literals are not scanned for symbolics. Whenever an &name within quotes in the PARM field is not defined, the ampersand is treated as if a double ampersand were coded and handled as a literal.

If cataloged procedures contain symbolic parameters, the installation should provide you with a list of the symbolic parameters used, what meaning is associated with each symbolic parameter, and what default value has been assigned to each of the symbolic parameters on the PROC statement. (The PROC statement is optional for cataloged procedures; therefore, there may be no default values assigned to the

symbolic parameters used in a cataloged procedure.) You need this information to determine what the symbolic parameter represents and to decide whether to use the default value or to assign a value to the symbolic parameter on the EXEC statement that calls the procedure.

To assign a value to a symbolic parameter, you code on the EXEC statement that calls the procedure:

symbolic parameter=value

Omit the ampersand that precedes the symbolic parameter. For example, if the symbolic parameter &NUMBER appears on a DD statement in the procedure, code NUMBER=value on the EXEC statement that calls the procedure. Any value you assign to a symbolic parameter is in effect only during the current execution of the procedure.

There are some things you should keep in mind as you assign values to symbolic parameters:

1. The value you assign can be any length, but it cannot be continued onto another statement.

2. If the value contains special characters, enclose the value in apostrophes (the enclosing apostrophes are not considered part of the value). If the special characters include apostrophes, each must be shown as two consecutive apostrophes.

3. If, on the EXEC statement, you assign more than one value to a symbolic parameter, the first value encountered is used.

4. The total length of a value assigned to a symbolic parameter cannot exceed 120 characters. This total includes the first through the last parameters, subparameters, or values in the card image.

5. If the symbolic parameter is a positional parameter followed by other parameters in the statement, it should be delimited in the procedure by a period instead of a comma. Then, if the parameter is nullified on the PROC statement or on an EXEC statement calling the procedure, the statement containing the symbolic parameter will not begin with a comma. The system recognizes the period as a delimiter; the period does not appear in the statement when you nullify or assign a value to the symbolic parameter. When you do assign a value to a symbolic parameter that is a positional parameter, you should follow the value with a comma; the value must then be enclosed in apostrophes since a comma is a special character.

   For example, in the following DD statement contained in a cataloged procedure named EXAMPLE, &POSPARM represents a positional parameter.

   ```
   //DEFINE DD &POSPARM.DSN=ATLAS,DISP=OLD
   ```

   To replace the symbolic parameter &POSPARM with the parameter DUMMY, you would code on the EXEC statement calling the procedure:

   ```
   //DOTHIS EXEC EXAMPLE,POSPARM='DUMMY,'
   ```

   When the cataloged procedure named EXAMPLE is executed, the DD statement named DEFINE appears as:

   ```
   //DEFINE DD DUMMY,DSN=ATLAS,DISP=OLD
   ```

   Note: Do not confuse positional parameters with positional subparameters. For a list of the positional parameters you can code on the DD statement, see "Positional and Keyword Parameters" in the section on the DD statement.

NULLIFYING A SYMBOLIC PARAMETER

Besides assigning values to symbolic parameters, you can nullify a symbolic parameter, i.e., tell the system to ignore the symbolic parameter.

To nullify a symbolic parameter, code on the EXEC statement that calls the procedure:

symbolic parameter=

Omit the ampersand that precedes the symbolic parameter in the procedure and do not follow the equal sign with a value.

For example, if a DD statement in a procedure named TIMES is

```
//DD8 DD UNIT=1403,UCS=&UCSINFO
```

and you want to nullify the symbolic parameter &UCSINFO, you would code:

```
//CALL EXEC TIMES,UCSINFO=
```

Example of Assigning Values to Symbolic Parameters

1. The following are the first four statements of a cataloged procedure named ASSEMBLE that contains symbolic parameters. The PROC statement assigns a default to the symbolic parameter &OBJECT and nullifies the symbolic parameter &LIST. Notice that the symbolic parameter &DEPT is not assigned a value on the PROC statement; therefore, the job step that calls this procedure must assign a value to &DEPT.

```
//DEF     PROC OBJECT=NODECK,LIST=
//ASM     EXEC PGM=IEUASM,PARM=('LINECNT=50',                    X
//             &LIST.LIST,&OBJECT)
//SYSLIB  DD   DSNAME=SYS1.MACLIB,DISP=OLD
//        DD   DSNAME=LIBRARY.&DEPT.MACS,DISP=OLD
```

When you call this procedure, you can assign values to the symbolic parameters by coding:

```
//STEP3  EXEC  ASSEMBLE,DEPT=D82,OBJECT=DECK
```

The value assigned to &OBJECT in this EXEC statement overrides the value assigned to &OBJECT in the PROC statement. Since no value is assigned to &LIST in this EXEC statement, LIST is nullified -- because that is the default specified in the PROC statement.

While the procedure is being executed, the first four statements of this procedure would appear as shown below.

```
//DEF     PROC OBJECT=NODECK,LIST=
//ASM     EXEC PGM=IEUASM,PARM=('LINECNT=50',                    X
//             LIST,DECK)
//SYSLIB  DD   DSNAME=SYS1.MACLIB,DISP=OLD
//        DD   DSNAME=LIBRARY.D82MACS,DISP=OLD
```

The above example applies to in-stream procedures as well as cataloged procedures. However, you must refer to the name on the PROC statement of the in-stream procedure when calling the procedure.

2. The following is an in-stream procedure that contains symbolic parameters. The PROC statement marks the beginning of the in-stream procedure and in this example assigns defaults to symbolic parameters &D, &U, &V, and &S. The procedure is named INSTREAM.

```
//INSTREAM     PROC D='(NEW,PASS)',U=2311,V='SER=66655',
//                  S='(TRK,(1,1,1))'
//IN1          EXEC PGM=IEWL,PARM='XREF,LIST,NCAL'
//SYSPRINT     DD   SYSOUT=A
//SYSUT1       DD   DSNAME=UTC,DISP=OLD,UNIT=2311,
//                  VOLUME=SER=66651
//SYSLIN       DD   DSNAME=UTE,DISP=OLD,UNIT=2311,
//                  VOLUME=SER=66652
//SYSLMOD      DD   DSNAME=&&LOAD,DISP=&D,UNIT=&U,
//                  VOLUME=&V,SPACE=&S
//            PEND
```

When you call this procedure, you must code the name on the PROC statement on the EXEC statement. You can assign values to the symbolic parameters by coding:

```
//CALL  EXEC INSTREAM,D='(NEW,PASS)',V='SER=66653'
```

The values assigned to &D and &V in this EXEC statement override the values assigned to these symbolic parameters in the PROC statement.

Since no value is assigned to &U OR &S, the defaults specified on the PROC statement are used when the procedure is executed.

While the procedure is being executed, it would appear as shown below.

```
//INSTREAM     PROC D='(NEW,PASS)',U=2311,V='Ser=66655',
//                  S=(TRK,(1,1,1))
//IN1          EXEC PGM=IEWL,PARM='XREF,LIST,NCAL'
//SYSPRINT     DD   SYSOUT=A
//SYSUT1       DD   DSNAME=UTC,DISP=OLD,UNIT=2311,
//                  VOLUME=SER=66651
//SYSLIN       DD   DSNAME=UTE,DISP=OLD,UNIT=2311,
//                  VOLUME=SER=66652
//SYSLMOD      DD   DSNAME=&&LOAD,DISP=(NEW,PASS),UNIT=2311,
//                  VOLUME=SER=66653,SPACE=(TRK,(1,1,1))
```

The PEND statement is necessary but is not executed.

3. The following are the first four statements of a cataloged procedure named TEST that contains symbolic parameters. The PROC statement nullifies the symbolic parameter &DUM1 and assigns a default value to the symbolic parameter &DUM2.

```
//TEST     PROC DUM1=,DUM2='DUMMY,'
//STEP1    EXEC PGM=IEFBR14
//DD1      DD   &DUM1.DSN=ABLE,DISP=OLD
//DD2      DD   &DUM2.DSN=BAKER,DISP=OLD
```

&DUM1 and &DUM2 are positional parameters. They are delimited by a period so that, if they are nullified, the DD statement does not begin with a comma. The system recognizes the period as a delimiter; the period does not appear in the statement when you nullify or assign a value to the symbolic parameter. The value assigned to DUM2 in the PROC statement is followed by a comma so that a comma will delimit the value when it appears in the statement in the procedure. The value must be enclosed in apostrophes because the comma is a special character.

When you call this procedure, you can reverse the default values on the PROC statement so that the DD statement named DD1 defines a dummy data set and the DD statement named DD2 defines an existing data set:

```
//STEPUP  EXEC TEST,DUM1='DUMMY,',DUM2=
```

The value assigned to &DUM1 on the EXEC statement overrides the nullification of &DUM1 on the PROC statement. &DUM2 is nullified on this EXEC statement, so the value assigned to &DUM2 on the PROC statement is ignored.

While this procedure is being executed, the first four statements will appear as shown below:

```
//TEST    PROC DUM1=,DUM2='DUMMY,'
//STEP1   EXEC PGM=IEFBR14
//DD1     DD   DUMMY,DSN=ABLE,DISP=OLD
//DD2     DD   DSN=BAKER,DISP=OLD
```

Overriding, Adding, and Nullifying Parameters on an EXEC Statement

You can override, add, or nullify parameters coded on EXEC statements contained in a cataloged or in-stream procedure. You make these changes on the EXEC statement that calls the procedure. You should override parameters only when you want to change their values. Do not override parameters to correct syntactical errors in the procedure. You cannot change the PGM parameter. The changes you make are in effect during the current execution of the procedure.

OVERRIDING EXEC STATEMENT PARAMETERS

To override an EXEC statement parameter in a procedure, identify on the EXEC statement that calls the procedure the parameter you are overriding, the name of the EXEC statement on which the parameter appears, and the change to be made. The format required to override a parameter is:

```
parameter.procstepname=change
```

For example, if one of the EXEC statements in the procedure named FILL is:

```
//STEP3  EXEC  PGM=DEF,REGION=100K
```

and you want to change REGION=100K to REGION=80K, you would code:

```
//CALL  EXEC  FILL,REGION.STEP3=80K
```

You can change more than one EXEC statement parameter in the procedure. For example, if one of the EXEC statements in the procedure name JKW is:

```
//STEP2  EXEC  PGM=OUT,TIME=(2,30),REGION=120K
```

and you want to change TIME=(2,30) to TIME=4 and REGION=120K to REGION=200K, you would code:

```
//STEP3  EXEC  JKW,TIME.STEP2=4,REGION.STEP2=200K
```

If you want to change different parameters that appear on different EXEC statements in the procedure, you must code all overriding parameters for one procedure step before those for the next step. For example, if the first three EXEC statements in a procedure named DART are:

```
//STEP1 EXEC PGM=JCTSB,PARM='*14863',REGION=100K
//STEP2 EXEC PGM=JCTRC,REGION=80K
//STEP3 EXEC PGM=JCTQD,COND=(8,LT),TIME=3
```

You want to make the following modifications:

1. Override the PARM parameter on the first EXEC statement.
2. Override the REGION parameter on the first EXEC statement.
3. Override the REGION parameter on the second EXEC statement.
4. Override the TIME parameter on the third EXEC statement.

The EXEC statement that calls the procedure would appear as:

```
//STEPC   EXEC DART,PARM.STEP1='*86348',                          X
//             REGION.STEP1=120K,REGION.STEP2=100K,               X
//             TIME.STEP3=(4,30)
```

You can code an EXEC statement parameter and omit the term "Procstepname." When you do this, the procedure is modified as follows:

- If the PARM parameter is coded, it applies only to the first procedure step. If a PARM parameter appears in a later EXEC statement, it is nullified.
- If the TIME parameter is coded, it applies to the total procedure. If the TIME parameter appears on any of the EXEC statements in the procedure, it is nullified.
- If any other parameter is coded, it applies to every step in the procedure. If the parameter appears on an EXEC statement, it is overridden; if the parameter does not appear on an EXEC statement, it is added.

For example, assume the EXEC statements in a procedure named RYIN are:

```
//STEP1   EXEC PGM=SECT,PARM=140947,REGION=100K
//STEP2   EXEC PGM=PARA,PARM=105600,COND=EVEN
//STEP3   EXEC PGM=SENT,PARM=L1644,REGION=80K
```

You want to make the following modifications to the procedure:

1. Override the PARM parameter in the first procedure step, and nullify all other PARM parameters in the procedure.
2. Assign the same region size to all steps in the procedure.

The EXEC statement that calls the procedure would appear as:

```
//SPAA    EXEC RYIN,PARM=L1644,REGION=136K
```

While the procedure named RYIN is being executed, these three EXEC statements would appear as:

```
//STEP1   EXEC PGM=SECT,PARM=L1644,REGION=136K
//STEP2   EXEC PGM=PARA,COND=EVEN,REGION=136K
//STEP3   EXEC PGM=SENT,REGION=136K
```

ADDING EXEC STATEMENT PARAMETER

To add a parameter to an EXEC statement in the procedure, identify on the EXEC statement that calls the procedure the parameter you are adding, the name of the EXEC statement to which you want to add the parameter, and the value you are assigning to the parameter. The format required to add a parameter is:

```
parameter.procstepname=value
```

Parameters you are adding and overriding for a step must be coded before those parameters you are adding and overriding for the next step.

For example, if the first two EXEC statements of a procedure named GLEAN are:

```
//STEP1   EXEC PGM=FAC,COND=(8,LT)
//STEP2   EXEC PGM=UP,PARM=377685
```

You want to make the following modifications to the procedure:

1. Override the COND parameter on the first EXEC statement.

2. Add the ROLL parameter to the first EXEC statement.

3. Add the REGION parameter to the second EXEC statement.

The EXEC statement that calls the procedure would appear as:

```
//STPA    EXEC GLEAN,COND.STEP1=(12,LT),                          X
//             ROLL.STEP1=(NO,NO),REGION.STEP2=88K
```

NULLIFYING EXEC STATEMENT PARAMETERS

To nullify a parameter on an EXEC statement in the procedure, identify, on the EXEC statement that calls the procedure, the parameter you want to nullify and the name of the EXEC statement on which the parameter appears. The format required to nullify a parameter is:

```
parameter.procstepname=
```

Parameters that you are nullifying, overriding, and adding to a step must be coded before those for the next step.

For example, if the first two EXEC statements of a procedure named GINN are:

```
//STEP1   EXEC PGM=INV,PARM='146,899',RD=R
//STEP2   EXEC PGM=DET,PARM=XYA34,COND=(80,GT)
```


You want to make the following modifications to the procedure:

1. Nullify the PARM parameter on the first EXEC statement.

2. Add the COND parameter to the first EXEC statement.

3. Override the COND parameter on the second EXEC statement.

The EXEC statement that calls the procedure would appear as:

```
//STEPY   EXEC GINN,PARM.STEP1=,COND.STEP1=(25,EQ),               X
//             COND.STEP2=(80,GE)
```

Examples of Overriding, Adding, and Nullifying Parameters on an EXEC Statement

1. You want to call the following cataloged procedure named ESEAP:

```
//STEPA   EXEC PGM=FLIER,PARM=7121190,ACCT=(4805,UNASGN)
//DDA     DD   DSNAME=LIBRARY.GROUP67,DISP=OLD
//DDB     DD   DSNAME=STAND.FIVE,DISP=OLD
//STEPB   EXEC PGM=VERSE,DPRTY=(11,13),PARM=780684,RD=R
//DDC     DD   UNIT=2311,SPACE=(TRK,(10,2))
//DDD     DD   DSNAME=COL.DISP=OLD
//DDE     DD   DDNAME=IN
```

You want to make the following modifications to the procedure:

1. Add the REGION parameter to both EXEC statements.
2. Add the DPRTY parameter to the first EXEC statement.
3. Override the ACCT parameter on the first EXEC statement.
4. Nullify the RD parameter on the second EXEC statement.
5. Add the COND parameter to the second EXEC statement.

The EXEC statement that calls the procedure would appear as:

```
//MINC    EXEC ESEAP,REGION=86K,DPRTY.STEPA=(11,13),                X
//             ACCT.STEPA=(4805,7554),RD.STEPB=,COND.STEPB=(60,LE)
```

The two EXEC statements in the procedure would appear as shown below while the procedure is being executed. These modifications do not appear on an output listing.

```
//STEPA   EXEC PGM=FLIER,PARM=7121190,ACCT=(4805,7554),             X
//             REGION=86K,DPRTY=(11,13)
//STEPB   EXEC PGM=VERSE,DPRTY=(11,13),REGION=86K,COND=(60,LE)
```

2. You want to call the following in-stream procedure named INLINE:

```
//INLINE  PROC
//STEP1   EXEC PGM=COMP,ACCT=(7037,2361),REGION=86K
//DD1     DD   DSNAME=INFORM,DISP=OLD,UNIT=2311,VOLUME=SER=75250
//DD2     DD   DSNAME=LCJWC,DISP=OLD,UNIT=2311,VOLUME=SER=76250
//STEP2   EXEC PGM=CHECKS,PARM=212334,COND=(50,LE),ACCT=(2001,0539)
//DD3     DD   DSNAME=PAY,DISP=OLD,UNIT=2311,VOLUME=SER=MEMORY
//DD4     DD   DSNAME=INCREAS,DISP=OLD,UNIT=2311,VOLUME=SER=33333
//        PEND
```

You want to make the following modifications to the procedure:

1. Add DPRTY parameter to both EXEC statements.
2. Nullify the REGION parameter on the first EXEC statement.
3. Override the ACCT parameter on the second EXEC statement.

The EXEC statement that calls the procedure would appear as:

```
//CALLER  EXEC INLINE,DPRTY=(11,13),REGION.STEP1=,                  X
//             ACCT.STEP2=(4710,5390)
```

The two EXEC statements in the procedure would appear as shown below while the procedure is being executed. These modifications do not appear on an output listing.

```
//STEP1   EXEC PGM=COMP,ACCT=(7037,2361),DPRTY=(11,13)
//STEP2   EXEC PGM=CHECKS,PARM=212334,COND=(50,LE),DPRTY=(11,13)
//             ACCT=(4710,5390)
```

Overriding, Adding, and Nullifying Parameters on a DD Statement

You can override, add, or nullify parameters coded on a DD statement contained in a cataloged procedure. You make these changes at the time the procedure is called; these changes are in effect during the current execution of the procedure. Use one DD statement to override, add, and nullify parameters on the same DD statement in the procedure.

OVERRIDING DD STATEMENT PARAMETERS

To override a parameter on a DD statement in the procedure, you must include a DD statement following the EXEC statement that calls the procedure. The ddname of this DD statement must identify the DD statement that contains the parameter you are overriding and the procedure step in which the DD statement appears. Code, in the operand field of this DD statement, the parameter you are overriding and the change; or code a mutually exclusive parameter that is to take the place of a parameter. Since mutually exclusive keywords are allowed during override processing, the first reference to such a keyword nullifies all further references, regardless of their positions on the DD statements. The format required for a DD statement following the EXEC statement is:

```
//procstepname.ddname     DD   parameter=change
                        or
//procstepname.ddname     DD   mutually exclusive parameter=value
```

For example, if one of the DD statements in a procedure step named STEP4 is:

```
//DD2 DD DSNAME=ABIN,DISP=OLD,VOLUME=SER=54896,UNIT=2400
```

and you want to change UNIT=2400 to UNIT=180, you would code:

```
//STEP4.DD2  DD  UNIT=180
```

When you code a mutually exclusive parameter on an overriding DD statement, the system replaces the parameter on the specified DD statement with the mutually exclusive parameter. For example, the parameters SYSOUT and DISP are mutually exclusive parameters. If one of the DD statements in a procedure step named PRINT is:

```
//DD8  DD  SYSOUT=C
```

and you do not want the data set printed, you could code:

```
//PRINT.DD8  DD  DISP=(NEW,DELETE)
```

You have replaced the SYSOUT parameter with the DISP parameter.

You can change more than one parameter that appears on a DD statement in the procedure. For example, if one of the DD statements in a procedure step named STEP5 is:

```
//DDX DD DSNAME=FIES,DISP=OLD,UNIT=2400-2,VOLUME=REF=*.STEP2.DDC
```

and you want this DD statement to define a new data set, you would code:

```
//STEP5.DDX  DD  DSNAME=RVA1,DISP=(NEW,KEEP)
```

If you want to change parameters that appear on different DD statements in the same procedure step, the overriding DD statements must be in the same order in the input stream as the corresponding DD statements in the procedure step. For example, if the first step of a procedure named AJG is:

```
//STEP1   EXEC PGM=MGR,REGION=80K
//DD1     DD   DSNAME=LONE,DISP=(NEW,DELETE),                    X
//            UNIT=2400,VOLUME=SER=568998
//DD2     DD   UNIT=TAPE
//DD3     DD   UNIT=2311,DISP=(,PASS),SPACE=(TRK,(20,2))
```

You want to make the following modifications to the procedure:

1. Change the UNIT parameter on the first DD statement.

2. Change the VOLUME parameter on the first DD statement.

3. Change the SPACE parameter on the third DD statement.

The statements in the input stream would appear as:

```
//CATP          EXEC AJG
//STEP1.DD1     DD   UNIT=2400-3,VOLUME=SER=WORK18
//STEP1.DD3     DD   SPACE=(CYL,(4,1))
```

If you want to change parameters that appear in different procedure steps in the cataloged procedure you are calling, the overriding DD statements must be in the same order as are the procedure steps.

The DCB parameter: If you want to change some of the keyword subparameters in the DCB parameter, you need not recode the entire DCB parameter. Instead, code only those subparameters that you are changing and any mutually exclusive subparameters that are to replace particular subparameters. For example, if one of the DD statements in a procedure step named NED is:

```
//DD3     DD   DSNAME=PER,DISP=(,KEEP),UNIT=2311,SPACE=(TRK,(88,5)),  X
//             DCB=(BUFNO=1,BLKSIZE=80,RECFM=F,BUFL=80)
```

and you want to change BLKSIZE=80 to BLKSIZE=320 and BUFL=80 to BUFL=320, you would code:

```
//NED.DD3   DD  DCB=(BLKSIZE=320,BUFL=320)
```

The DCB subparameters BUFNO and RECFM remain unchanged.

When you are overriding a procedure DD statement that contains a DCB parameter and the overriding DD statement uses a backward reference to copy the DCB information on an earlier DD statement, the DCB information on the procedure DD statement overrides any of the corresponding subparameters that are copied. For example, if one of the DD statements in a step named NED of a procedure named CATROC is:

```
//DD5     DD   DSNAME=PER,UNIT=2311,SPACE=(TRK,(88,5)),              X
//            DCB=(BLKSIZE=640,RECFM=FB)
```

and you have in your input stream:

```
//STP1    EXEC PGM=A
//DD1     DD   DSN=AIR,UNIT=2311,SPACE=(TRK,(10,1)),                 X
//            DCB=(BLKSIZE=320,RECFM=FBA,BUFL=320)
//STEP2   EXEC CATROC
//NED.DD5 DD   DCB=*.STP1.DD1
```

The DD statement DD5 in a cataloged procedure would appear as shown below while the procedure is being executed. (The DCB information on the procedure statement overrides any of the corresponding subparameters that are copied.) This modification does not appear on output listing.

```
//DD5      DD   DSNAME=PER,UNIT=2311,SPACE=(TRK,(88,5)),          X
//             DCB=(BLKSIZE=640,RECFM=FB,BUFL=320)
```

If you want to override a DD statement that contains a dsname positional subparameter in the DCB parameter, you must recode the dsname subparameter, even though you do not want to change it. For example, if one of the DD statements in a procedure step named BANK is:

```
//DD5      DD   DSNAME=SAVE,DISP=(NEW,KEEP),UNIT=2311,           X
//             SPACE=(CYL,(12,2)),DCB=(ACCNT,BUFNO=5,KEYLEN=2)
```

and you want to change BUFNO=5 to BUFNO=3, you would code:

```
//BANK.DD5      DD   DCB=(ACCNT,BUFNO=3)
```

Both the dsname ACCNT and KEYLEN subparameters remain unchanged. You must code ACCNT on the overriding DD statement.

ADDING DD STATEMENT PARAMETERS

To add a parameter to a DD statement in the procedure, you must include a DD statement following the EXEC statement that calls the procedure. The ddname of this DD statement must identify the DD statement to which you are adding a parameter and the procedure step in which the DD statement appears. Code, in the operand field of this DD statement, the parameter you are adding. The format required for a DD statement following the EXEC statement is:

```
//procstepname.ddname  DD  parameter=value
```

For example, if one of the DD statements in a procedure step named STPTWO is:

```
//DDM      DD   DSNAME=TYPE,DISP=(,KEEP),UNIT=2400
```

and you want to add the VOLUME parameter, you would code:

```
//STPTWO.DDM  DD  VOLUME=SER=569433
```

If you want to add parameters or change parameters that appear on different DD statements, the overriding DD statements must be in the same order in the input stream as the corresponding DD statements in the procedure.

NULLIFYING DD STATEMENT PARAMETERS

There may be parameters on a DD statement that you do not want to override, but you want the system to ignore. Also, when you modify a DD statement in a procedure by overriding certain parameters or adding parameters, there may be some parameters remaining that no longer have meaning for your data set definition but would effect processing of the data set. To temporarily remove these parameters, you can nullify them. (If you are replacing a parameter with a mutually exclusive parameter, do not nullify the parameter that is being replaced.)

To nullify a parameter on a DD statement in the procedure, you must include a DD statement following the EXEC statement that calls the procedure. The ddname of this DD statement must identify the DD statement that contains the parameter you are nullifying and the procedure step in which the DD statement appears. Code in the operand field of this DD statement the parameter you are nullifying followed by an equal sign; do not follow the equal sign with a value. The format required for a DD statement following the EXEC statement is:

```
//procstepname.ddname DD parameter=
```

For example, if one of the DD statements in a procedure step named SALLS is:

```
//DDP      DD   DSNAME=STEP,DISP=OLD,UNIT=2314,                    X
//             VOLUME=SER=556978
```

and you are overriding the DSNAME, DISP, and UNIT parameters, adding the DCB parameter, and want the VOLUME parameter ignored, you would code:

```
//SALLS.DDP    DD   DSNAME=&&TEMP,DISP=(,PASS),UNIT=2400-2,        X
//                  DCB=(DEN=2,TRTCH=ET),VOLUME=
```

To nullify the DCB parameter, each DCB subparameter must be nullified individually. For example, if a DD statement contains DCB=(RECFM=FBA,BLKSIZE=160,LRECL=80), then DCB=(RECFM=,BLKSIZE=,LRECL=) must be coded on the overriding DD statement in order to nullify the DCB parameter.

To nullify a DUMMY parameter, code the DSNAME parameter on the overriding DD statement, but do not use the data set name NULLFILE. (Coding DSNAME=NULLFILE has the same effect as coding the DUMMY parameter.)

Caution: When you are overriding a procedure DD statement that contains the SPACE parameter and the overriding DD statement defines an existing data set, be sure to nullify the SPACE parameter. When a secondary quantity is coded on the procedure DD statement, the system uses this value to assign additional space to the data set instead of the secondary quantity you may have specified when the data set was created. Also, the RLSE subparameter, when specified on the procedure statement, causes the system to release any of the existing data set's unused space.

If you want to nullify, add, or override parameters that appear on different DD statements, the overriding DD statements must be in the same order in the input stream as the corresponding DD statements in the procedure.

Examples of Overriding, Adding, and Nullifying Parameters on a DD Statement

1. You want to call the following procedure named SALL:

```
//STP1     EXEC PGM=GLR14
//DD11     DD   DSNAME=XTRA.LEVEL,DISP=OLD
//DD12     DD   DSNAME=CONDS,DISP=(,PASS),UNIT=2400
//DD13     DD   DUMMY,DSNAME=LAST,VOLUME=REF=*.DD11,DISP=(,CATLG)
//STP2     EXEC PGM=FAIR
//DD21     DD   DSNAME=*.STP1.DD12,DISP=(OLD,DELETE)
//DD22     DD   DSNAME=JFTZ,DISP=(NEW,KEEP),UNIT=2311,             X
//             SPACE=(CYL,(2,1),RLSE)
//DD23     DD   SYSOUT=G
```

You want to modify the procedure as follows:

1. Change the data set name on the statement named DD12 from CONDS to C8495.

2. Add the VOLUME parameter to the statement named DD12.

3. Nullify the DUMMY parameter on the statement named DD13.

4. Change the disposition on the statement named DD21 from DELETE to KEEP.

5. Define an existing data set on the statement named DD22.

6. Add the parameter UNIT on the statement named DD23.

7. Add the parameter SPACE on the statement named DD23.

The EXEC statement that calls the procedure and the overriding DD statements that follow it would appear as:

```
//CALL          EXEC SAIL
//STP1.DD12     DD   DSNAME=C8495,VOLUME=SER=979354
//STP1.DD13     DD   DSNAME=SOMENAME
//STP2.DD21     DD   DISP=(OLD,KEEP)
//STP2.DD22     DD   SPACE=,DSNAME=GR1833,DISP=OLD,                    X
//                   VOL=SER=577632
//STP2.DD23     DD   UNIT=2314,SPACE=(TRK,(150,15))
```

The cataloged procedure would appear as shown below while the procedure is being executed. These modifications do not appear on an output listing.

```
//STP1     EXEC PGM=GLF14
//DD11     DD   DSNAME=XTRA.LEVEL,DISP=OLD
//DD12     DD   DSNAME=C8495,DISP=(,PASS),UNIT=2400,                  X
//             VOLUME=SER=979354
//DD13     DD   DSNAME=LAST,VOL=REF=*.DD11,DISP=(,CATLG)
//STP2     EXEC PGM=FAIR
//DD21     DD   DSNAME=*.STP1.DD12,DISP=(OLD,KEEP)
//DD22     DD   DSNAME=GR1833,DISP=OLD,UNIT=2311,VOL=SER=577632
//DD23     DD   SYSOUT=G,UNIT=2314,SPACE=(TRK,(150,15))
```

2. You want to call the following in-stream procedure named CARDS:

```
//CARDS    PROC
//STEPA    EXEC PGM=FIGURE
//DDA1     DD   DSNAME=NUMBERS,DISP=OLD
//DDA2     DD   DSNAME=PROCESS,DISP=(,PASS),UNIT=2311,
//             SPACE=(TRK,(1,1,1))
//STEPB    EXEC PGM=RESULT
//DDB1     DD   DSNAME=VSC,DISP=OLD
//DDB2     DD   DSNAME=*.STEPA.DDA2,DISP=(OLD,KEEP)
//DDB3     DD   SYSOUT=C
//         PEND
```

You want to modify the procedure as follows:

1. Change the data set name on the DDA1 statement from NUMBERS to NAMES.
2. Add the VOLUME parameter to the DDA2 statement.
3. Add the parameters UNIT and SPACE on the DDB3 statement.

The EXEC statement that calls the procedure and the overriding DD statements that follow it would appear as:

```
//CALL          EXEC CARDS
//STEPA.DDA1    DD   DSNAME=NAMES
//STEPA.DDA2    DD   VOLUME=SER=5858
//STEPB.DDB3    DD   UNIT=2311,SPACE=(TRK,(150,15))
```

The in-stream procedure would appear as shown below while the procedure is being executed. These modifications do not appear on an output listing. The PROC statement is processed only when it contains symbolic parameters.

```
//STEPA    EXEC PGM=FIGURE
//DDA1     DD   DSNAME=NAMES,DISP=OLD
//DDA2     DD   DSNAME=PROCESS,DISP=(,PASS),UNIT=2311,
//             SPACE=(TRK,(1,1,1)),VOLUME=SER=5858
//STEPB    EXEC PGM=RESULT
//DDB1     DD   DSNAME=VSC,DISP=OLD
//DDB2     DD   DSNAME=*.STEPA.DDA2,DISP=(OLD,KEEP)
//DDB3     DD   SYSOUT=C,UNIT=2311,SPACE=(TRK,(150,15))
```

Overriding DD Statements that Define Concatenated Data Sets

When a concatenation of data sets is defined in a cataloged procedure and you attempt to override the concatenation with one DD statement, only the first (named) DD statement is overridden. To override others, you must include an overriding DD statement for each DD statement; the DD statements in the input stream must be in the same order as the DD statements in the procedure. The second and subsequent overriding statements must not be named. If you do not wish to change one of the concatenated DD statements, leave the operand field blank on the corresponding DD statement in the input stream. (This is the only case where a blank operand field for a DD statement is valid.)

For example, suppose you are calling a procedure that includes the following sequence of DD statements in STEPC:

```
//DD4      DD   DSNAME=A.B.C,DISP=OLD
//        DD   DSNAME=STEP,DISP=OLD,UNIT=2311,VOL=SER-X12182
//        DD   DSNAME=TYPE3,DISP=OLD,UNIT=2311,VOLUME=SER=BL142
//        DD   DSNAME=A.B.D,DISP=OLD
```

If you want to override the DD statements that define the data sets named STEP and A.B.D, the sequence of DD statements in the input stream would appear as:

```
//STEPC.DD4    DD
//            DD   DSNAME=INV.CLS,DISP=OLD
//            DD
//            DD   DSNAME=PAL8,DISP=OLD,UNIT=2311,VOL=SER=125688
```

Adding DD Statements to a Procedure

You can add DD statements to a procedure when you call the procedure. These additional DD statements are in effect only while the procedure is being executed.

To add a DD statement to a procedure step, follow the EXEC statement that calls the procedure and any overriding DD statements for that step with the additional DD statement. The format of a DD statement to be added to a procedure step is:

```
//procstepname            DD   parameters
                     or
//procstepname.ddname     DD   parameters
```

You must identify the procedure step to which the data set is to be added in the name of the DD statement. If you also assign a ddname, it must be different from all the other ddnames in the procedure step.

If you do not assign a ddname, the data set is concatenated to the data set defined by the preceding DD statement. If the data set is the first to be added to the procedure step, it is concatenated to the last data set defined in the procedure step.

For example, the first step of a cataloged procedure named MART is:

```
//STEP1   EXEC PGM=DATE
//DDM     DD   VOLUME=BPS(MFMG),DISP=OLD,                          X
//            UNIT=2311,VOLUME=SER=554982
//DDN     DD   UNIT=SYSQE
```

You want to make the following modifications to the procedure:

1. Change the UNIT parameter on the DD statement named DDM;
2. Concatenate a data set to the data set defined by DDN;
3. Add another data set with DDNAME=DDO.

The statements in the input stream would appear as:

```
//PROC          EXEC MART
//STEP1.DDM     DD   UNIT=180
//STEP1.DDN     DD
//              DD   DSNAME=SPEC,DISP=OLD,UNIT=2311
//STEP1.DDO     DD   UNIT=181
```

Examples of Adding DD Statements to a Procedure

1. You want to call the following procedure named D995A:

```
//SA      EXEC PGM=ANALY
//DDA1    DD   DSNAME=PROJ.0843,DISP=OLD
//DDA2    DD   DDNAME=SYSIN
//DDA3    DD   SYSOUT=B
//SB      EXEC PGM=MANMO3
//DDB1    DD   UNIT=2400
//DDB2    DD   UNIT=2400
//DDB3    DD   DSNAME=X54,VOLUME=SER=(36544,36545),
//             UNIT=(2400,2),DISP=(OLD,KEEP)
```

You want to modify the procedure as follows:

1. Supply the data set definition for the DDA2 statement by adding a DD statement.

2. Change the SYSOUT parameter on the DDA3 statement to UNIT=1403.

3. Add a DD statement to the step named SB.

The EXEC statement that calls the procedure and the overriding and additional DD statements that follow it would appear as:

```
//PROCED        EXEC D995A
//SA.DDA3       DD   UNIT=1403,DISP=NEW
//SA.SYSIN      DD   *
                .
                .
                .
                data
                .
                .
/*
//SB.DDB4       DD   UNIT=(2400,,SEP=(DDB1,DDB2))
                .
```

The cataloged procedure would appear as shown below while the procedure is being executed. These modifications do not appear on output listing.

```
//SA       EXEC PGM=ANALY
//DDA1     DD   DSNAME=PROJ.0843,DISP=OLD
//DDA2     DD   *
//DDA3     DD   UNIT=1403,DISP=NEW
//SB       EXEC PGM=MANMO3
//DDB1     DD   UNIT=2400
//DDB2     DD   UNIT=2400
//DDB3     DD   DSNAME=X54,VOLUME=SER=(36544,36545),                X
//             UNIT=(2400,2),DISP=(OLD,KEEP)
//DDB4     DD   UNIT=(2400,,SEP=(DDB1,DDB2))
```

2. You want to call the following in-stream procedure named WORK:

```
//WORK     PROC
//STP1     EXEC PGM=PROD
//DD1      DD   DSNAME=PROJECT,DISP=OLD
//DD2      DD   DDNAME=SYSIN
//        PEND
```

You want to modify the procedure by supplying the data set definition for the DD2 statement by adding a DD statement.

The EXEC statement that calls the procedure and the additional DD statement that follows it would appear as

```
//ADD          EXEC WORK
//STP1.SYSIN   DD   *
                .
                .
                .
                data
                .
                .
                .
                .
/*
```

The in-stream procedure would appear as shown below while the procedure is being executed. These modifications do not appear on the output listing.

```
//STP      EXEC PGM=PROD
//DD1      DD   DSNAME=PROJECT,DISP=OLD
//DD2      DD   *
```

Writing Procedures: Cataloged and In-Stream

Why Catalog Job Control Statements

Applications performed at your installation on a regular basis and applications that require many control statements can be simplified when the control statements for these applications are cataloged. Once the job control statements for an application are cataloged on the procedure library, any programmer who wants to perform the application need only submit a JOB and EXEC statement. On the EXEC statement, he refers the system to the control statements required to perform the application. If there are modifications the programmer wants to take for the duration of the job step, he assigns values to symbolic parameters on the EXEC statement and follows the EXEC statement with overriding DD statements.

Why Use In-Stream Procedures

In-stream procedures appear within the job stream instead of in the procedure library. Like cataloged procedures, they eliminate the necessity of repeating the same set of control statements in a job. An in-stream procedure can be executed any number of times during a job in which it appears and fifteen uniquely named in-stream procedures can appear in one job. In-stream procedures can be modified just as cataloged procedures. They also provide you with a means of testing procedures before adding them to the procedure library as cataloged procedures. Because an in-stream procedure may exist in the form of cards, it can be considered a "portable procedure" in that it can easily be moved from one input stream to another.

THE CONTENTS OF CATALOGED AND IN-STREAM PROCEDURES

Cataloged and in-stream procedures contain one or more EXEC statements, each followed by associated DD statements. Each EXEC statement identifies the program to be executed, and the DD statements that follow define the input, output, and work data sets to be used by the program. Each EXEC statement and its associated DD statements are called a procedure step.

Cataloged and in-stream procedures cannot contain:

1. EXEC statements that refer to other cataloged procedures.
2. JOB, delimiter, or null statements.
3. DD statements with the ddname JOBLIB.
4. DD statements with * or DATA coded in the operand field.

A cataloged or in-stream procedure can contain a DD statement with the ddname STEPLIB. If a procedure step requires use of a program in a private library other than SYS1.LINKLIB, you define that library on this DD statement. If the DD statement is not overridden when the procedure is called, it makes the private library available to the step. (For information on the STEPLIB DD statement, see the chapter "Special Ddnames" in Section IV of this publication.)

For ease in modifying a cataloged or in-stream procedure, you can include symbolic parameters in the procedure. How to use symbolic parameters is described next.

USING SYMBOLIC PARAMETERS IN A PROCEDURE

A symbolic parameter is characterized by a name preceded by an ampersand (&) and appears in the operand field of a cataloged procedure statement. A symbolic parameter stands for a parameter, a subparameter, or a value.

Symbolic parameters allow a programmer who calls the procedure to e easily modify the procedure for the duration of the job step. When the programmer calls the procedure, he assigns values to the symbolic parameters on the EXEC statement. When you prepare control statements that you plan to catalog, you can include a PROC statement and assign default values to any of the symbolic parameters that are included. When you prepare control statements to be used as an in-stream procedure, you must include a PROC statement which can be used to assign default values to any of the symbolic parameters that are included.

A symbolic parameter is one to seven alphameric and national (#,@,$) characters preceded by a single ampersand. The first character must be alphabetic or national. Since a single ampersand defines a symbolic parameter, you code double ampersands when you are not defining a symbolic parameter. For example, if you want to pass 543&LEV to a processing program by means of the PARM parameter on an EXEC statement, you must code PARM='543&&LEV'. The system treats the double ampersands as if a single ampersand has been coded, and only one ampersand appears in the results.

The following are examples of symbolic parameters:

```
//STEP1 EXEC PGM=COB,PARM='P1,&P2,P3'

//DD1 DD DSNAME=&&FIX,UNIT=&DEVICE,SPACE=(CYL,(&SPACE,10))

//DD2 DD DSNAME=&&CHAG,UNIT=2400,DCB=BLKSIZE=&LENGTH
```

Keyword parameters that can be coded on an EXEC statement cannot be used to define symbolic parameters. For example, &PGM and &REGION cannot be used as symbolic parameters. The system will not recognize a symbolic parameter if you enclose it in apostrophes, unless it is part of the PARM parameter.

Any parameter, subparameter, or value in the procedure that may vary each time the procedure is called is a good candidate for definition as a symbolic parameter. For example, if different values can be passed to a processing program by means of the PARM parameter on one of the EXEC statements, you might define the PARM parameter field as one or more symbolic parameters, PARM=&ALLVALS or PARM=&DECK&CODE.

If symbolic parameters are defined in the cataloged or in-stream procedures used at your installation, the definitions should be consistent. For example, every time the programmer is to assign his department number to a symbolic parameter, no matter which procedure he is calling, the symbolic parameter could be defined as &DEPT. In different procedures you could code ACCT=(43877,&DEPT) and DSNAME=LIBRARY.&DEPT.MACS. The programmer would assign his department number on the EXEC statement that calls the procedure whenever &DEPT appears in a procedure. Of course, in order for the programmer to know that he is to assign his department number to the symbolic parameter &DEPT, the installation must make this information available to all the programmers that may be using the cataloged procedures.

You can define two or more symbolic parameters in succession without including a comma to delimit the symbolic parameters, for example, &P1&P2. You can also define a portion of a parameter, subparameter, or value as a symbolic parameter. You do this by placing the symbolic parameter before, after, or in between the information that is not variable.

If you place a symbolic parameter after some information that does not vary, it is not necessary to code a delimiter. The system recognizes a symbolic parameter when it encounters the single ampersand.

If you place a symbolic parameter before some information that does not vary, a period may be required following the symbolic parameter to distinguish the end of the symbolic parameter and the beginning of the information that does not vary. A period is required following the symbolic parameter when:

1. The character following the symbolic parameter is an alphabetic, numeric, or national character.

2. The character following the symbolic parameter is a left parenthesis or a period.

In these cases, the system recognizes the period as a delimiter, and the period does not appear after a value is assigned to the symbolic parameter. (A period will appear after a value is assigned to the symbolic parameter when two consecutive periods are coded.)

The following examples are valid ways of combining symbolic parameters and information that does not vary.

Placing a symbolic parameter after information that does not vary:

1. LIBRARY(&MEMBER)

2. USERLIB.&LEVEL

Placing a symbolic parameter before information that does not vary:

1. '&OPTION+15'

2. &PASS.A43B8

The period is required because an alphabetic character follows the symbolic parameter.

3. &URNO.54328

The period is required because a numeric character follows the symbolic parameter.

4. &LIBRARY.(MEMG)

The period is required because a left parenthesis follows the symbolic parameter.

5. &FILL..GROUP5

A period is to appear in the results; therefore, two consecutive periods are coded.

When a value is assigned to the symbolic parameter, this value and the parameter, subparameter, or value that this is a portion of cannot exceed 120 characters.

The programmer who calls a procedure assigns values to the symbolic parameters contained in the procedure. He can also nullify symbolic parameters. A delimiter, such as a leading comma or a trailing comma, next to a symbolic parameter is not automatically removed when the symbolic parameter is nullified. For example, if the operand field contains VOLUME=SER=(111111,&KEY), the comma preceding &KEY is not removed when &KEY is nullified. If the symbolic parameter that is nullified is a positional parameter, a comma must remain to indicate its absence. In other cases, a delimiter that is not removed when the symbolic parameter is nullified may cause a syntax error. To help the programmer who nullifies a symbolic parameter avoid this error condition, define those symbolic parameters that may be nullified without the delimiter. For example, you could code VOLUME=SER=(111111&KEY). The delimiter is included when a value is assigned to the symbolic parameter. For example, the programmer would code KEY=',222222'.

A cataloged or in-stream procedure statement may utilize DDNAME and DCB parameters to define data in the input stream. Such a statement should not contain symbolic parameters when the automatic SYSIN batching reader is used. (Information on the cataloged procedure for the automatic SYSIN batching reader is contained in the MFT and MVT Guides publication.

The PROC statement. When establishing cataloged or in-stream procedures that contain symbolic parameters it is generally good practice to assign default values to the symbolic parameters. These default values are used if the programmer who calls the procedure does not assign values to one or more of the symbolic parameters.

You assign default values on a PROC statement. The PROC statement is optional in cataloged procedures; if it is used, the PROC statement must be the first statement in the procedure. The PROC statement is described in Section X of this publication. The PEND statement which is used to mark the end of an in-stream procedure is described in Section IX.

ADDING AND MODIFYING CATALOGED PROCEDURES

You add procedures to the procedure library by using the IEBUPDTE utility program. You also use this utility program to permanently modify existing procedures. How to use this utility program for adding and modifying cataloged procedures is described in the chapter "The IEBUPDTE Program" in the Utilities publication.

When you add a cataloged procedure to the procedure library, that procedure cannot be executed before the job that adds it to the procedure library terminates. When you modify an existing cataloged procedure, the operator must be notified. What the operator must do before he allows the job to be executed is described in the chapter "How to Run Jobs That Update System Data Sets" in the Operator's Reference publication.

Appendix B: Using the Restart Facilities

When a job step abnormally terminates, you may have to resubmit the job for execution. This means lost computer time and a delay in obtaining the desired results. To reduce these effects, you can use the restart facilities.

If a job step abnormally terminates or if a system failure occurs, the restart facilities allow you to request that the job step be restarted either at the beginning of the step (step restart) or within the step (checkpoint restart). Furthermore, restart can occur automatically after abnormal termination, or it can be deferred until the job is resubmitted.

Restarts

For automatic step restart to occur, the RD parameter must request it on the JOB statement or on the EXEC statement associated with the step that abnormally terminates. (The RD parameter on the JOB statement is described in Section II of this publication; the RD parameter on the EXEC statement is described in Section III.) Automatic checkpoint restart can occur only if a CHKPT macro instruction is executed in the processing program prior to abnormal termination.

If restart is deferred until the job is resubmitted, the RESTART parameter must be coded on the JOB statement of the resubmitted job. (The RESTART parameter is described in Section II of this publication.) The RESTART parameter identifies the step or the step and the checkpoint at which execution is to be resumed. A deferred restart may be initiated regardless of how the resubmitted job was previously terminated (normally or abnormally) and regardless of whether an automatic restart occurred during the original execution.

AUTOMATIC STEP RESTART

If an abnormally terminated step is to be automatically restarted, the RD parameter must be coded as RD=R or RD=RNC. Execution resumes at the beginning of the abnormally terminated step.

AUTOMATIC CHECKPOINT RESTART

After an automatic checkpoint restart, execution resumes at the instruction immediately following the last CHKPT macro instruction that was successfully executed in the abnormally terminated step. An automatic checkpoint restart cannot occur if you suppress the action of the CHKPT macro instruction; you do this by coding RD=NC or RD=RNC. Also, an automatic checkpoint restart cannot occur if you code RD=NR; however, RD=NR allows the CHKPT macro instruction to establish a checkpoint.

DEFERRED STEP RESTART

To perform a deferred step restart, the RESTART parameter must identify the step at which execution is to be resumed. Steps preceding the restart step are interpreted but are not initiated.

Since dispostion processing occurred during the original execution of the job, you may have to modify control statements associated with the restart step before you resubmit the job. Modifications may be required in two cases:

1. A data set was defined as NEW during the original execution. If it was created during the original execution, you must change the data set's status to OLD, define a new data set, or delete the data set before resubmitting the job.

2. A data set was passed and was to be received by the restart step or a step following the restart step. If the passed data set is not cataloged, you must supply, in the receiving step, volume serial numbers, device type, data set sequence number, and label type. (Label type cannot be retrieved from the catalog.)

To limit the number of modifications required before you resubmit the job, you can assign conditional dispositions during the original execution. (Data sets assigned a temporary name or no name can only be assigned a conditional disposition of DELETE.) If deferred step restart will be performed, conditional dispositions should be used:

- To delete all new data sets created by the restart step.

- To keep all old data sets used by the restart step, other than those passed to the step. (If a nontemporary data set is defined as DISP=(OLD,DELETE), it is very important that you assign a conditional disposition of KEEP.)

- To catalog all data sets passed from steps preceding the restart step to the restart step or to steps following the restart step.

Additional changes can be made to your control statements before resubmitting the job. For example, you can vary device and volume configurations and request step restart on an alternate system with the same configuration as used originally. You can also make changes to your data.

DEFERRED CHECKPOINT RESTART

To perform a deferred checkpoint restart, the RESTART parameter must identify the step and the checkpoint at which execution is to be resumed. The SYSCHK DD statement, which defines the checkpoint data set, must also be included. (The SYSCHK DD statement is described in the chapter "Special Ddnames" in Section IV.)

An internal representation of your statements is kept as control information within the system. Some of the control information for the restart step or steps following the restart step may have to be modified before execution can be resumed at a checkpoint. The following modifications for the restart step are automatically made by the system, using information contained in the checkpoint entry:

- The status of data sets used by the step is changed from NEW to OLD. (If a new data set was assigned a nonspecific volume and had not been opened before the checkpoint was established, this change is not made.)

- If nonspecific volumes were requested for a data set used in the restart step, the assigned device type and volume serial numbers are made part of the control information.

- For a multivolume data set, the volume being processed when the checkpoint was established is mounted.

The only required modification that you must make to a control statement is to supply certain information about a data set that was being passed by a step preceding the restart step to a step following the restart step. You must supply, in the receiving step, volume serial numbers, device type, data set sequence number, and label type. You will not have to make these modifications if, during the original execution, you assigned a conditional disposition of CATLG to such data sets. If the data is cataloged, the system can retrieve this information from the catalog. (Label type cannot be retrieved from the catalog.) You should also use conditional dispositions to keep all data sets used by the restart step. Data sets assigned a temporary name or no name can only be assigned a conditional disposition of DELETE. Therefore, if you plan a deferred checkpoint restart, you should not define your data sets as temporary. (For any nontemporary data set that may be deleted, it is very important that you assign a conditional disposition of KEEP.)

Before resubmitting the job for checkpoint restart, you can make other modifications to control statements associated with the restart step or steps following the restart step. The following items apply to the step in which restart is to occur:

- The DD statements in the restart step can be altered, but the statements must have the same names as used originally. You can also include additional DD statements.

- If a data set was open at the time a checkpoint was established and restart is to begin at that checkpoint, DD statements in the restart step can define the same data set. If there is no need to process a data set after restart, you can define the data set by coding the DUMMY parameter or DSNAME=NULLFILE on a DD statement provided that: (1) the basic sequential access method (BSAM) or the queued sequential access method (QSAM) was being used to process the data set when the checkpoint was established, (2) the data set is not the checkpoint data set that is being used to restart the job step, and (3) the job step is not restarted from a checkpoint that was established in an end-of-volume exit routine for the data set. The name of the DD statement must be the same as the one used for the data set during the original execution of your program.

- If DUMMY is not specified, the DD statements must define the same data sets. Also, the data sets must not have been moved on the volume or onto another volume.

- If a data set was not open when the checkpoint was established and is not needed during restart, you can replace the parameters used to define the data set with the DUMMY parameter.

- You can alter the data in the restart step. If you omit the data, a delimiter statement (/*) is not required, unless the data was preceded by a DD DATA statement.

Modifications you might want to make to control statements following the restart step are: varying device and volume configurations, altering data, and possibly, requesting checkpoint restart on an alternate system with the same configuration as used originally. If the parameters PGM, COND, SUBALLOC, and VOLUME=REF refer to steps preceding the restart step, you must resolve these references before resubmitting the job. (A backward reference of VOLUME=REF is allowed if the referenced statement includes VOLUME=SER=(serial number).)

Examples of Using the Restart Facilities

1. The following control statements illustrate the preparations that would be made for either an automatic step or checkpoint restart before the job is submitted for the first time.

```
//STMRG3   JOB  54321,A.USER,MSGLEVEL=(1,0),RD=R
//STEP1    EXEC PGM=SIMPSOPT
//INPUT    DD   DSN=SORTIN,VOL=SER=100468,UNIT=2400,                X
//             DISP=(OLD,DELETE)
//OUTPUT   DD   DSN=INV(+1),UNIT=2311,VOL=SER=555334,              X
//             SPACE=(3200,(200,100)),DISP=(NEW,CATLG)
//WORK1    DD   UNIT=2400,DISP=(NEW,DELETE)
//WORK2    DD   UNIT=2400,DISP=(NEW,DELETE)
//CHKPT    DD   UNIT=2400,DISP=(NEW,DELETE)
//STEP2    EXEC PGM=MYMERGE
//MERG1    DD   DSN=INV(+1),DISP=OLD
//MERG2    DD   DSN=M5,VOL=SER=(092501,092502,092503),
//             UNIT=(2400,3),DISP=(OLD,KEEP)

//RESULTS  DD   DSN=M6,UNIT=2400,VOL=SER=(100101,100102,           X
//             100103),DISP=(NEW,KEEP)
```

Here, the RD parameter requests step restart for any abnormally terminated job step. In STEP1, the DD statement CHKPT defines a checkpoint data set. For this step, once a CHKPT macro instruction is executed, only automatic checkpoint restart is performed. An automatic checkpoint restart cannot occur in STEP2 since a checkpoint data set is not defined.

2. The following control statements illustrate the preparations that would be made for either an automatic or deferred step restart before the job is submitted for the first time.

```
//STMRG3   JOB  54321,A.USER,MSGLEVEL=(1,0),RD=R
//STEP1    EXEC PGM=SIMPSORT
//INPUT    DD   DSN=SORTIN,VOL=SER=100468,UNIT=2400,                X
//             DISP=(OLD,DELETE,KEEP)
//OUTPUT   DD   DSN=INV(+1),UNIT=2311,VOL=SER=555334,              X
//             SPACE=(3200,(200,100)),DISP=(NEW,CATLG,DELETE)

//WORK1    DD   UNIT=2400,DISP=(NEW,DELETE)
//WORK2    DD   UNIT=2400,DISP=(NEW,DELETE)
//STEP2    EXEC PGM=MYMERGE
//MERG2    DD   DSN=M5,VOL=SER=(092501,(092502,092503),            X
//             UNIT=(2400,3),DISP=(OLD,KEEP)
//RESULTS  DD   DSN=M6,UNIT=2400,VOL=SER=(100101,100102,           X
//             100103),DISP=(NEW,KEEP,DELETE)
```

If you are resubmitting this job for step restart, you must code the RESTART parameter on the JOB statement and identify the step at which execution is to be resumed. If execution is to be resumed with STEP2, the MERG1 DD statement must be changed to refer to the generation data set by means of its present relative generation number, i.e., DSN=INV(0).

3. The following control statements illustrate the preparations that would be made for an automatic step or checkpoint restart or a deferred checkpoint restart before the job is submitted for the first time.

```
//STMRG3   JOB  54321,A.USER,MSGLEVEL=(1,0),RD=R
//STEP1    EXEC PGM=SIMPSORT
//INPUT1   DD   DSN=SORTIN,VOL=SER=100468,UNIT=2400,                X
//              DISP=(OLD,DELETE,KEEP)
//OUTPUT   DD   DSN=INV(+1),UNIT=2311,VOL=SER=555334,               X
//              SPACE=(3200,(200,100)),DISP=(NEW,CATLG,KEEP)
//WORK1    DD   DSN=A,UNIT=2400,DISP=(NEW,DELETE,CATLG)
//WORK2    DD   DSN=B,UNIT=2400,DISP=(NEW,DELETE,CATLG)
//CHKPT    DD   DSN=O,UNIT=2400,DISP=(NEW,DELETE,CATLG)
//STEP2    EXEC PGM=MYMERGE
//MERG1    DD   DSN=INV(+1),DISP=OLD
//MERG2    DD   DSN=M5,VOL=SER=(092501,092502,092503),              X
//              UNIT=(2400,3),DISP=(OLD,KEEP)
//RESULTS  DD   DSN=M6,UNIT=2400,VOL=SER=(100101,100102,           X
//              100103),DISP=(NEW,KEEP)
```

Either an automatic checkpoint restart or a deferred checkpoint restart can occur in STEP1 if the step abnormally terminates. To perform a deferred checkpoint restart, the RESTART parameter must be coded on the JOB statement and a SYSCHK DD statement must be included before resubmitting the job. Only automatic step restart can occur in STEP2. The data sets that would normally be defined as temporary have been defined as nontemporary data sets so conditional dispositions can be assigned to them.

Appendix C: Creating and Retrieving Indexed Sequential Data Sets

Indexed sequential (ISAM) data sets are created and retrieved using special subsets of DD statement parameters and subparameters. Each data set can occupy up to three different areas of space:

1. Prime area -- This area contains data and related track indexes. It exists for all indexed sequential data sets.

2. Overflow area -- This area contains overflow from the prime area when new data is added. It is optional.

3. Index area -- This area contains master and cylinder indexes associated with the data set. It exists for any indexed sequential data set that has a prime area occupying more than one cylinder.

Indexed sequential data sets must reside on direct access volumes. The data set can reside on more than one volume and the device types of the volumes may in some cases differ.

Creating an Indexed Sequential Data Set

One to three DD statements can be used to define a new indexed sequential data set. When you use three DD statements to define the data set, each DD statement defines a different area and the areas must be defined in the following order:

1. Index area.

2. Prime area.

3. Overflow area.

When you use two DD statements to define the data set, the areas must be defined in the following order:

1.	Index area.	1.	Prime area
		or	
2.	Prime area.	2.	Overflow area.

When you use one DD statement to define the data set, you are defining the prime area and, optionally, the index area.

When more than one DD statement is used to define the data set, assign a ddname only to the first DD statement; the name field of the other statements must be blank.

The only DD statement parameters that can be coded when defining a new indexed sequential data set are the DSNAME, UNIT, VOLUME, LABEL, DCB, DISP, SPACE, SEP, and AFF parameters. When to code each of these parameters and what restrictions apply are described in the following paragraphs.

THE DSNAME PARAMETER

The DSNAME parameter is required on any DD statement that defines a new temporary or nontemporary indexed sequential data set. To identify the area you are defining, you follow the DSNAME parameter with the area: DSNAME=name(INDEX), DSNAME=name(PRIME), or DSNAME=name(OVFLOW). If you are using only one DD statement to define the data set, code DSNAME=name(PRIME) or DSNAME=name.

When reusing previously allocated space to create an ISAM data set, the DSNAME parameter must contain the name of the old data set to be overlaid.

THE UNIT PARAMETER

The UNIT parameter is required on any DD statement that defines a new indexed sequential data set unless VOLUME=REF=reference is coded. You must request a direct access device in the UNIT parameter and must not request DEFER.

If there are separate DD statements defining the prime and index areas, you must request the same number of direct access devices for the prime area as there are volumes specified in the VOLUME parameter. You may request only one direct access volume for an index area and one for an overflow area.

A DD statement for the index area or overflow area can request a device type different than the type requested on the other statements.

Another way to request a device is to code UNIT=AFF=ddname; where the named DD statement requests the direct access device or device type you want. New direct access data sets imply deferred mounting. In this case, do not code UNIT=AFF.

THE VOLUME PARAMETER

The VOLUME parameter is required only if you want an area of the data set written on a specific volume or if the prime area requires use of more than one volume. (If the prime area and index area are defined on the same statement, you cannot request more than one volume on the DD statement. When a multiple volume prime area is required specific serial numbers must be specified.) Either supply the volume serial number or numbers in the VOLUME parameter or code VOLUME=REF=reference.

If you make a nonspecific volume request and the first volume searched does not have enough space available, the step may abnormally terminate, depending on the following circumstances:

- If you requested multiple volumes, space allocation is not fulfilled and the job step is terminated.
- If you requested a single volume and the DD statement containing the request is the first or only DD statement defining the data set, the system attempts to find another volume with sufficient space available.
- If you requested a single volume and the DD statement containing the request is the second or third DD statement defining the data set, space allocation is not fulfilled and the job step is terminated.

With both specific and nonspecific volume requests, you can also request a private volume with the PRIVATE subparameter and request that

the private volume be retained with the RETAIN subparameter. These subparameters are discussed in detail in the chapter on the VOLUME parameter.

Note: If a new ISAM data set is being created with a nonspecific volume request and it's DSNAME already exists on a volume(s) eligible for allocation, the job may fail due to duplicate names on the volume. Successful allocation, under these conditions, depends upon where the old data set resides in relation to eligible devices. Failures of this type can be corrected by either scratching the old data set or renaming the new data set before resubmitting the job.

THE LABEL PARAMETER

The LABEL parameter need only be coded to specify a retention period (EXPDT or RETPD) or password protection (PASSWORD).

THE DCB PARAMETER

The DCB parameter must be coded on every DD statement that defines an indexed sequential data set. At minimum, the DCB parameter must contain DSORG=IS or ORG=ISU. Other DCB subparameters can be coded to complete the data control block if it has not been completed by the processing program. When more than one DD statement is used to define the data set, code all the DCB subparameters on the first DD statement. Code DCB=*.ddname on the remaining statement or statements; ddname is the name of the DD statement that contains the DCB subparameters.

When reusing previously allocated space and recreating an ISAM data set, desired changes in the DCB parameter must be coded on the DD statement. Although you are creating a new data set, some DCB subparameters cannot be changed if you want to use the space that the old data set used. The DCB subparameters, you can change are: BFALN, BLKSIZE, CYLOFL, DSORG, HIARCHY, KEYLEN, LRECL, NCP, NTM, OPTCD, RECFM, and RKP.

THE DISP PARAMETER

If you are creating a new data set and not reusing preallocated space, the DISP parameter need only be coded if you want to keep, DISP=(,KEEP), catalog, DISP=(,CATLG), or pass, DISP=(,PASS), the data set. If you are reusing previously allocated space and recreating an ISAM data set, code DISP=OLD. The newly created data set will overlay the old one.

In order to catalog the data set when DISP=(,CATLG) is coded or pass the data set when DISP=(,PASS) is coded, the data set must be defined on only one DD statement. If the data set was defined on more than one DD statement and the volumes on which the data set now resides correspond to the same device type, you can use the IEHPROGM utility program to catalog the data set. Refer to the chapter "The IEHPROGM Program" in the Utilities publication for details.

THE SPACE PARAMETER

Nonspecific Allocation Technique

The SPACE parameter is required on any DD statement that defines a new indexed sequential data set. Use either the recommended nonspecific allocation technique or the more restricted absolute track (ABSTR) technique. If more than one DD statement is used to define the data set, all must request space using the same technique.

You must request the primary quantity in cylinders (CYL). When the DD statement that defines the prime area requests more than one volume, each volume is assigned the number of cylinders requested in the SPACE parameter.

One of the subparameters of the SPACE parameter, the "index" subparameter, is used to indicate how many cylinders are required for an index. When one DD statement is used to define the prime and index areas and you want to explicitly state the size of the index, code the "index" subparameter.

The CONTIG subparameter can be coded in the SPACE parameter. However, if CONTIG is coded on one of the statements, it must be coded on all of them.

You cannot request a secondary quantity for an indexed sequential data set. Also, you cannot code the subparameters RLSE, MXIG, ALX, and ROUND.

Absolute Track Technique

The number of tracks you request must be equal to one or more whole cylinders. The address of the beginning track must correspond with the first track of a cylinder other than the first cylinder on the volume. When the DD statement that defines the prime area requests more than one volume, space is allocated for the prime area beginning at the specified address and continuing through the volume and onto the next volume until the request is satisfied. (This can only be done if the volume table of contents of the second and all succeeding volumes is contained within the first cylinder of each volume.)

One of the subparameters of the SPACE parameter, the "index" subparameter, is used to indicate how many tracks are required for an index. The number of tracks specified must be equal to one or more cylinders. When one DD statement is used to define the prime and index areas and you want to explicitly state the size of the index, code the "index" subparameter.

If you also specify a number of tracks or cylinders to be used as a directory or index, the primary quantity you code will be the total number of tracks or cylinders assigned. Example: if SPACE=(CYL,(10,1)) is specified, 10 cylinders will be assigned.)

Note: If the indexed sequential data set is to reside on more than one volume and an error is encountered as the volumes are being allocated to the data set, follow this procedure before resubmitting the job: use the IEHPROGM utility program to scratch the data set labels on any of the volumes to which the data set was sucessfully allocated. This utility program is described in the chapter "The IEHPROGM Program" in the Utilities publication.

THE SEP OR AFF PARAMETER

The SEP or AFF parameter is coded only if you want channel separation from the area or areas defined on the preceding statement or statements in the group. In order for the areas to be written using separate channels, you must also request devices by their actual address, e.g., UNIT=190.

Area Arrangement of an Indexed Sequential Data Set

When you create an indexed sequential data set, the arrangement of the areas is based on two criteria:

1. The number of DD statements used to define the data set.
2. What area each DD statement defines.

An additional criterion is used when you do not include a DD statement that defines the index area:

3. Is an index size coded in the SPACE parameter of the DD statement that defines the prime area?

Figure 40 illustrates the different arrangements that can result based on the criteria listed above. In addition, Figure 40 indicates what restrictions apply on the number and types of devices that can be requested.

CRITERIA			RESTRICTIONS ON DEVICE TYPES AND NUMBER OF DEVICES REQUESTED.	RESULTING ARRANGEMENT OF AREAS
1.Number of DD statements	2.Area defined on a DD statement	3.Index size coded?		
3	INDEX PRIME OVFLOW	-	None	Separate index, prime, and overflow areas.
2	INDEX PRIME	-	None	Separate index and prime areas.[1]
2	PRIME OVFLOW	No	None	Separate prime and overflow areas. An index area is at the end of the overflow area.
2	PRIME OVFLOW	Yes	The statement defining the prime area cannot request more than one device.	Separate prime and overflow areas. An index area is embedded in the prime area.
1	PRIME	No	None	Prime area with index area at its end.[2]
1	PRIME	Yes	Cannot request more than one device.	Prime area with embedded index area.

[1]If both areas are on volumes that correspond to the same device type, an overflow area is established if one of the cylinders allocated for the index area is only partially used. The overflow area is established in the unused portion of that cylinder.
[2]If the unused portion of the index area is less than one cylinder, it is used as an overflow area.

Figure 40. Area Arrangement of Indexed Sequential Data Sets

Retrieving an Indexed Sequential Data Set

If all areas of an existing indexed sequential data set reside on volumes of the same device type, you can retrieve the entire data set with one DD statement. If the index or overflow resides on a volume of a different device type, you must use two DD statements. If the index and overflow reside on volumes of different device types, you must use three DD statements to retrieve the data set. The DD statements are coded in the following order:

1. First DD statement - defines the index area

2. Second DD statement - defines the prime area

3. Third DD statement - defines the overflow area

The only DD statement parameters that can be coded when retrieving an indexed sequential data set are the DSNAME, UNIT, VOLUME, DCB, and DISP parameters. When to code each of these parameters and what restrictions apply are described in the following paragraphs.

THE DSNAME PARAMETER

The DSNAME parameter is always required. If the data set was passed from a previous step, identify it by a backward reference.

THE UNIT PARAMETER

The UNIT parameter is required unless the data set resides on one volume and was passed or unless the data set was cataloged. You identify in the UNIT parameter the device type and how many of these devices are required.

If the data set resides on more than one volume and the volumes correspond to the same device type, you need only one DD statement to retrieve the data set. Request one device in the UNIT parameter per volume. If the index or overflow area of the data set resides on a different type of volume than the other areas, you must use two DD statements to retrieve the data set. On one DD statement, request the device type required to retrieve the index or overflow area. On the other DD statement, request the device type and the number of devices required to retrieve the prime area and the overflow area if the overflow area resides on the same device type. If the index and the overflow areas reside on different device types from the prime area, a third DD statement is needed.

THE VOLUME PARAMETER

The VOLUME parameter is required unless the data set resides on one volume and was passed or unless the data set was cataloged. Identify in the VOLUME parameter the serial numbers of the volumes on which the data set resides. Code the serial numbers in the same order as they were coded on the DD statements used to create the data set.

THE DCB PARAMETER

The DCB parameter must be coded unless the data set was passed from a previous step. The DCB parameter must always contain DSORG=IS or DSORG=ISU. Other DCB subparameters can be coded to complete the data control block if it has not been completed by the processing program.

THE DISP PARAMETER

The DISP parameter must always be coded. The first subparameter of the DISP parameter must be MOD or OLD. You can, optionally, assign a disposition as the second subparameter.

Example of Creating and Retrieving an Indexed Sequential Data Set

1. The following job step includes the DD statements that could be used to create an indexed sequential data set. Each area of the indexed sequential data set is defined on a separate DD statement.

```
//OUTPUT4 EXEC PGM=INCLUDE
//GROUP1  DD   DSNAME=PART86(INDEX),DISP=(,KEEP),UNIT=2314,       X
//            VOLUME=SER=538762,SPACE=(CYL,10,,CONTIG),          X
//            DCB=(DSORG=IS,RECFM=F,LRECL=80,RKP=1,KEYLEN=8)
//        DD  DSNAME=PART86(PRIME),DISP=(,KEEP),UNIT=2311,2),    X
//            VOLUME=SER=(538763,538764),                        X
//            SPACE=(CYL,(25),,CONTIG),DCB=*.GROUP1
//        DD  DSNAME=PART86(OVFLOW),DISP=(,KEEP),UNIT=2311,      X

//            VOLUME=SER=538765,SPACE=(CYL,15,,CONTIG),          X
//            DCB=*.GROUP1
```

The following job step includes the DD statements required to retrieve the indexed sequential data set created above.

```
//INPUT12 EXEC PGM=ADD
//RET4    DD   DSNAME=PART86,DCB=DSORG=IS,UNIT=2314,             X
//            DISP=OLD,VOLUME=SER=538762
//        DD   DSNAME=PART86,DCB=DSORG=IS,UNIT=(2311,3),         X
//            DISP=OLD,VOLUME=SER=(538763,538764,838765)
```

Two DD statements are required to retrieve the data set because the index area resides on a volume of a different device type than the volumes on which the prime and overflow areas reside.

Appendix D: Creating and Retrieving Generation Data Sets

A generation data set is one of a collection of successive, historically related, cataloged data sets known as a generation data group. The system keeps track of each data set in a generation data group as it is created so that new data sets can be chronologically ordered and old ones easily retrieved.

To create or retrieve a generation data set, you identify the generation data group name in the DSNAME parameter and follow the group name with a relative generation number. When creating a generation data set, the relative generation number tells the system whether this is the first data set being added during the job, the second, the third, etc. When retrieving a generation data set, the relative generation number tells the system how many data sets have been added to the group prior to the current job.

A generation data group can consist of cataloged sequential, partitioned, indexed sequential (if the data set is defined on one DD statement), and direct data sets residing on tape volumes, direct access volumes, or both. Generation data sets can have like or unlike DCB attributes and data set organizations. If the attributes and organizations of all generations in a group are identical, the generations can be retrieved together as a single data set (up to 255 data sets can be retrieved in this way).

Before You Define the First Generation Data Set

Before you define the first generation data set, you must build a generation data group index. This index provides lower-level entries for as many generation data sets (up to 255) as you would like to have in your generation data group. The system uses these lower-level indexes to keep track of the chronological order of the generation data sets. The index must reside on the system residence volume, or an alternate control volume. You use the IEHPROGM utility program to build your index; this program is described in the chapter "The IEHPROGM Program" in the Utilities publication.

Another requirement of generation data groups is that a data set label must exist on the same volume as the index. The system uses this label to refer to DCB attributes when you define a new generation data set. There are two ways to satisfy this requirement: (1) create a model data set label before you define the first generation data set; or (2) use the DCB parameter to refer the system to an existing cataloged data set each time you define a new generation data set.

Creating a Model Data Set Label

To create a model data set label, you must define a data set and request that it be placed on the same volume as the generation data group index. This ensures that there is always a data set label on the same volume as the index to which the system can refer.

The name you assign to the data set may be the same or different than the name assigned to the generation data group. (If you assign the same name for both, the data set associated with the model data set label cannot be cataloged.) You may request a space allocation of zero tracks or cylinders. The DCB attributes you can supply are DSORG, RECFM, OPTCD, BLKSIZE, LRECL, KEYLEN, and RKP.

This is an example of creating a model data set label:

```
//DD1      DD   DSNAME=PAY.WEEK,DISP=(NEW,KEEP),UNIT=2311,          X
//             VOLUME=SER=SYSRES,SPACE=(TRK,0),DCB=(RECFM=FB,       X
//             LRECL=240,BLKSIZE=960)
```

You need not create a model data set label for every generation data group whose indexes reside on the same volume. Instead, you may create one model data set label to be used by any number of generation data groups. If you create only one model, you should not supply any DCB attributes. When you create a generation data set, you specify the name of the model in the DCB parameter and follow the name with a list of all the DCB subparameters required for the new generation data set, i.e., DCB=(dsname,list of attributes).

Referring the System to a Cataloged Data Set

If there is a cataloged data set that resides on the same volume as your generation data group index and you are sure that data set will exist as long as you are adding data sets to your generation data group, you need not create a model data set label. When you create a generation data set, you specify the name of the cataloged data set in the DCB parameter, i.e., DCB=dsname. If all the DCB attributes are not contained in the label of the cataloged data set, or if you want to override certain attributes, follow the data set name with these attributes, i.e., DCB=(dsname,list of attributes).

Creating a Generation Data Set

When defining a new generation data set, you always code the DSNAME, DISP, and UNIT parameters. Other parameters you might code are the VOLUME, SPACE, LABEL, and DCB parameters.

Note: If you make a nonspecific request for a tape volume for a new generation data set and the data set is never opened, the system will catalog the data set with a volume serial number of blanks. This prevents you from retrieving an incorrect generation and thus maintains the integrity of the generation data group.

Tape generation data sets that were never opened and that were cataloged may only be uncataloged by specifying the data set name parameter in the format of generation data group name and the relative generation number or by using IEHPROGM.

THE DSNAME PARAMETER

In the DSNAME parameter, you code the name of the generation data group followed by a number enclosed in parentheses. This number must be 1 or greater. If this is the first data set you are adding to a particular generation data group during the job, code +1 in parentheses. Each time during the job you add a data set to the same generation data group, increase the number by one.

Any time you refer to this data set later in the job, you use the same relative generation number as was used earlier. At the end of the job, the system updates the relative generation numbers of all generations in the group to reflect the additions.

Note: Do not code the DSNAME parameter using index (alias) names, even if they do exist for the highest level index.

Unpredictable results may occur if using a relative generation number which causes the actual generation number to exceed G9999.

THE DISP PARAMETER

New generations are assigned a status of NEW and a disposition of CATLG in the DISP parameter, i.e., DISP=(NEW,CATLG). If you do not specify a disposition, or specify a disposition other than CATLG, the system assumes CATLG.

THE UNIT PARAMETER

The UNIT parameter is required on any DD statement that defines a new generation data set unless VOLUME=REF=reference is coded. In the UNIT parameter, you identify the type and number of devices you want (tape or direct access).

Another way to request a device is to code UNIT=AFF=ddname; where the named DD statement requests the device or device type you want.

THE VOLUME PARAMETER

You may assign a volume in the VOLUME parameter or let the system assign one for you. The VOLUME parameter can also be used to request a private volume (PRIVATE), to retain the private volume (RETAIN), and to indicate that more volumes may be required (volume count).

THE SPACE PARAMETER

The SPACE parameter is coded only when the generation data set is to reside on a direct access volume. The SPLIT or SUBALLOC parameter can be coded in place of the SPACE parameter if the data set's organization permits the use of these parameters.

THE LABEL PARAMETER

You can specify label type, password protection (PASSWORD), and a retention period (EXPDT or RETPD) in the LABEL parameter. If the data set will reside on a tape volume and is not the first data set on the volume, specify a data set sequence number. The expriation date from the model data set control block is used unless an expiration date is coded on the DD statement.

THE DCB PARAMETER

A model data set label that has the same name as the group name may exist. If this is so, and if the label contains all the attributes required to define this generation, you need not code the DCB parameter. If all the attributes are not contained in the label, or if you want to override certain attributes, code these attributes in the DCB parameter, i.e., DCB=(list of attributes).

If a model data set label has a different name than the group name and if the label contains all the attributes required to define this generation data set, only the name of the data set associated with the model data set label need be coded. Code the name in the DCB parameter, i.e., DCB=dsname. If all the attributes are not contained in the label, or if you want to override certain attributes, follow the data set name with these attributes, i.e., DCB=(dsname,list of attributes).

If a model data set label does not exist, you must code the name of a cataloged data set that resides on the same volume as the generation data group index, i.e., DCB=dsname. If all the attributes are not contained in the label for this data set, or if you want to override certain attributes, follow the data set name with these attributes, i.e., DCB=(dsname,list of attributes).

Retrieving a Generation Data Set

To retrieve a generation data set, you always code the DSNAME and DISP parameters. Other parameters you might code are the UNIT, LABEL, and DCB parameters.

THE DSNAME PARAMETERS

In the DSNAME parameter, you code the name of the generation data group followed by a number enclosed in parentheses. The number coded is a relative generation number with zero (0) referring to the highest generation number present in the catalog prior to the start of the current job. A reference to the second highest generation number present, prior to job initiation, would be coded (-1). If you wish to refer to a new member that was created in an earlier step of this job code the same positive integer that was used when it was created.

When any data set member is referenced more than once in a job the later reference(s) will have the same relative generation number as the first reference, even if another generation has been added during the job.

Note: Relative generation numbers are based on the catalog, as it existed at job initiation, plus any changes made to it by cataloging new members of the data set during the job. If a member is uncataloged during the job the later use of that members, or earlier members, relative generation number(s) will result in incorrect data set references.

If you want to retrieve all generations of a generation data group by concatenating all existing data sets in the generation data group, starting with the most recent and ending with the oldest, with unit affinity to the most recent data set, you specify the generation data group name without a generation number, e.g., DSNAME=WEEKLY.PAYROLL. You can retrieve all generations as a single data set only if the attributes and organizations of all generations are identical.

THE DISP PARAMETER

The DISP parameter must always be coded. The first subparameter of the DISP parameter must be OLD, SHR, or MOD. You can, optionally, assign a disposition as the second subparameter. Using the disposition subparameter PASS can cause mishandling of a data set at job termination because a passed GDG cannot be received.

When retrieving all generations of a generation data group as a single data set, you should avoid coding PASS as the second subparameter. In all such retrievals the unit and volume information for each generation level will be obtained from the catalog and not via the pass mechanism.

THE UNIT PARAMETER

Code the UNIT parameter when you want more than one device assigned to the data set. Code the number of devices you want in the unit count subparameter, or, if the data set resides on more than one volume and you want as many devices as there are volumes, code P in place of the unit count subparameter.

THE LABEL PARAMETER

Code the LABEL parameter when the data set has other than standard labels. The expiration date from the model data set control block is used unless an expiration date is coded on the DD statement.

THE DCB PARAMETER

Code the DCB parameter when the data set has other than standard labels and DCB information is required to complete the data control block.

Resubmitting a Job for Restart

Certain rules apply when you refer to generation data sets in a job resubmitted for restart (the RESTART parameter is coded on the JOB statement).

For step restart: If step restart is performed, generation data sets that were created and cataloged in steps preceding the restart step must not be referred to in the restart step or in steps following the restart step by means of the same relative generation numbers that were used to create them. Instead, you must refer to a generation data set by means of its present relative generation number. For example, if the last generation data set created and cataloged was assigned a generation number of +2, it would be referred to as 0 in the restart step and in steps following the restart step. In this case, the generation data set assigned a generation number of +1 would be referred to as -1.

For checkpoint restart: If generation data sets created in the restart step were kept instead of cataloged (i.e., DISP=(NEW,CATLG,KEEP) was coded), you can, during checkpoint restart, refer to these data sets and generation data sets created and cataloged in steps preceding the restart step by means of the same relative generation numbers that were used to create them.

Reference:

1. Generation data sets can be created and retrieved using utility programs. How to do this is described in "Appendix E: Generation Data Groups" in the Utilities publication. Also described in this appendix is how to put indexed sequential data sets in a generation data group.

Example of Creating and Retrieving Generation Data Sets

1. The following job step includes the DD statements that could be used to add three data sets to a generation data group.

```
//STEPA   EXEC PGM=PROCESS
//DD1     DD   DSNAME=A.B.C(+1),DISP=(NEW,CATLG),UNIT=2400,      X
//             VOL=SER=13846,LABEL=(,SUL)
//DD2     DD   DSNAME=A.B.C(+2),DISP=(NEW,CATLG),UNIT=2311,      X
//             VOL=SER=10311
//DD3     DD   DSNAME=A.B.C(+3),DISP=(NEW,CATLG),UNIT=2301,      X
//             VOL=SER=28929,SPACE=(480,(150,20)),DCB=LRECL-120, X
//             BLKSIZE=480)
```

The first two DD statements do not include the DCB parameter; therefore, a model data set label must exist on the same volume as the generation data group index and must have the same name as the generation data group (A.B.C). Since the DCB parameter is coded on the third DD statement, the attributes LRECL and BLKSIZE, along with the attributes included in the model data set label, are used.

The following job includes the DD statements required to retrieve the generation data sets defined above when no other data sets have been added to the generation data group.

```
//JWC     JOB  CLASS=B
//STEP1   EXEC PGM=REPORT9
//DDA     DD   DSNAME=A.B.C(-2),DISP=OLD,LABEL=(,SUL)
//DDB     DD   DSNAME=A.B.C(-1),DISP=OLD
//DDC     DD   DSNAME=A.B.C(0),DISP=OLD
```

Appendix E: Default Parameter Values Supplied in the Input Reader Procedure

As your control statements are read and interpreted, the input reader assigns default values to specific parameters that are not coded and checks for violations of certain restrictions. The default values for specific parameters and the restrictions are specified in the cataloged procedure for the input reader.

The input reader is controlled by a reader/interpreter cataloged procedure supplied by IBM or the installation. The default parameter values and restrictions will probably differ in the IBM-supplied and the installation-supplied procedures.

How to Keep Track of the Default Values and Restrictions

Figure 40 lists the parameters for which default values are assigned when they are not coded on specific control statements. The default values assigned to these parameters when an IBM-supplied cataloged procedure is used are also listed. Space is left in the right-hand portion of the table so you can write in the default values that will be assigned when an installation-supplied procedure is used. Figure 40 also lists those restrictions that must be checked as the control statements are read and tells which apply when an IBM-supplied procedure is used. Space is left in the right-hand portion of the table so you can write in which of these restrictions apply when a installation-supplied procedure is used.

The page on which Figure 41 appears may be removed from the publication and placed in a convenient location, so that you and other programmers can refer to it.

Default Parameter Values

Parameter	Statement	IBM-Supplied	Installation-Supplied Name:	Installation-Supplied Name:
MSGCLASS	JOB	A		
MSGLEVEL	JOB	(0,1)		
PRTY	JOB	1		
REGION	JOB and EXEC	50K		
TIME	EXEC	30 minutes		
ROLL	JOB and EXEC	(YES,NO)		
UNIT (note 2)	DD	SYSDA		
SPACE (note 2)	DD	(TRK,(50,10))		
BLKSIZE (note 3)	DD	(note 1)		
BUFNO (note 3)	DD	(note 1)		

Restrictions

Parameter or Subparameter	Statement	IBM-Supplied	Installation-Supplied Name:	Installation-Supplied Name:
Accounting Information	JOB	not required		
Programmer's Name	JOB	not required		
BLP (note 4)	DD	NL assumed		

Notes:

1. The default value differs in each of the three IBM-supplied procedures, as follows:

	Procedure Name		
	RDR	RDR400	RDR3200
BLKSIZE	80	400	3200
BUFNO	2	2	1

2. The default values for the UNIT and SPACE parameters are used when you do not include these parameters on a DD statement that defines a data set being routed through an output stream (i.e., the SYSOUT parameter is coded on the DD statement). These default values also apply to data sets being routed through an output stream during any automatic restart.

3. The default values for the DCB subparameters BLKSIZE and BUFNO are used when you do not include these subparameters on a DD statement that defines data in the input stream (i.e., DD * or DD DATA statement).

4. BLP is a subparameter in the LABEL parameter that requests that tape label processing be bypassed.

Figure 41. Default Values and Restrictions Supplied in the Supplied in the Input Reader Procedures

Appendix F: A Checklist

When you create or retrieve a data set, the system requires certain information. This information is supplied on the DD statement that defines the data set.

This appendix can be used as a checklist: As you code your DD statements, find the function you are performing in the left-hand column of Figure 42. Across from the function are two separate lists of parameters. These parameters describe the information that you must supply to the system and the information that you may have to supply. You can compare your DD statement with what is listed to make sure all the required information is available to the system.

Following Figure 42 are examples of the DD statements that might be used when performing functions described in the table. Each example is keyed by number to a particular block within the table. If you do not understand why a parameter is listed for the function, either look at the example that corresponds to the number within the block or refer to the parameter description in Section IV of this publication.

FUNCTION: Creating a Data Set	Information That Is Always Required	Information That May Be Required
Temporary Data Sets		
Creating a Data Set on a Unit Record Device	UNIT 1	DCB 2 UCS
Creating a Data Set on a Tape Volume	UNIT 3	DCB 4 VOLUME LABEL
Creating a Data Set in the Output Stream	SYSOUT 5,6,7	DCB 8 UNIT SPACE FCB UCS
Creating A Data Set on a Direct Access Volume	UNIT 9 SPACE	DCB 10 VOLUME LABEL
Nontemporary Data Sets		
Creating a Data Set on a Tape Volume	UNIT 11 DSNAME DISP	LABEL 12 DCB VOLUME
Creating a Generation Data Set on a Tape Volume	DISP 13 UNIT DSNAME	DCB 14 LABEL VOLUME

Figure 42. A Checklist (Part 1 of 3)

Creating a Nontemporary Data Set (con't)	Information That Is Always Required	Information That May Be Required
Creating a Sequential Data Set on a Direct Access Volume (BSAM or QSAM)	UNIT 15 DSNAME DISP SPACE,SPLIT,orSUBALLOC	LABEL 16 DCB VOLUME
Creating a Data Set With Direct Organization on a Direct Access Volume (BDAM)	UNIT 17 DSNAME DISP SPACE or SUBALLOC DCB	LABEL 18 VOLUME
Creating a Partitioned Data Set on a Direct Access Volume (BPAM)	UNIT 19 DSNAME DISP SPACE or SUBALLOC	LABEL 20 VOLUME DCB
Creating a New Member for a Partitioned Data Set	DISP 21 DSNAME	UNIT 22 VOLUME
Creating a Data Set With Indexed Sequential Organization on a Direct Access Volume (QISAM)	UNIT 23 DSNAME DISP DCB SPACE	VOLUME 24 LABEL
Creating a Generation Data Set on a Direct Access Volume	SPACE 25 DISP UNIT DSNAME	DCB 26 LABEL VOLUME
FUNCTION: Retrieving a Data Set		
Retrieving a Cataloged Data Set	DSNAME 27 DISP	DCB 28 LABEL UNIT
Retrieving a Noncataloged Data Set on a Tape Volume	DSNAME 29 UNIT VOLUME DISP	LABEL 30 DCB
Retrieving a Noncataloged Sequential Data Set on a Direct Access Volume (BSAM or QSAM)	UNIT 31 VOLUME DSNAME DISP	LABEL 32
Retrieving a Noncataloged Data Set with Direct Organization on a Direct Access Volume (BDAM)	UNIT 33 VOLUME DSNAME DISP	LABEL 34
Retrieving a Member of a Partitioned Data Set (BPAM)	DISP 35 DSNAME	UNIT 36 VOLUME

Figure 42. A Checklist (Part 2 of 3)

Retrieving a Data Set	Information That Is Always Required	Information That May Be Required
Retrieving a Data Set With Indexed Sequential Organization on a Direct Access Volume (QISAM or BISAM)	DSNAME 37 UNIT VOLUME DCB DISP	
Retrieving a Passed Data Set	DSNAME 38 DISP	LABEL 39 DCB VOLUME UNIT
Retrieving a Generation Data Set	DSNAME 40 DISP	DCB 41 LABEL UNIT

Figure 42. A Checklist (Part 3 of 3)

Examples

```
1   //DDA    DD  UNIT=1404

2   //DDB    DD  UNIT=1403,UCS=PCAN,DCB=PRTSP=2

3   //DDC    DD  UNIT=2400

4   //DDD    DD  UNIT=2400-1,DCB=DEN=1,VOLUME=SER=14187,LABEL=2

5   //DDE    DD  SYSOUT=L

6   //DDF    DD  SYSOUT=G,DCB=PRTSP=2

7   //DDG    DD  SYSOUT=(M,,7956)

8   //DDH    DD  SYSOUT=B,UNIT=2301,SPACE=(1024,(75,25)),                X
                 DCB=BLKSIZE=2048

9   //DDI    DD  UNIT=SYSDA,SPACE=(TRK,(20,5))

10  //DDJ    DD  UNIT=2311,SPACE=(CYL,(2,1)),DCB=(RECFM=S,LRECL=X),      X
    //           LABEL=(,SUL),VOLUME=SER=190853

11  //DDK    DD  UNIT=2400,DSNAME=OUT,DISP=(NEW,KEEP)

12  //DDL    DD  UNIT=2400-2,DSNAME=WLK18,DISP=(,KEEP),LABEL=(,NL),      X
    //           DCB=TRTCH=C,VOLUME=SER=1540

13  //DDM    DD  DISP=(,CATLG),UNIT=2400,DSNAME=WEEK.PAY(+1)

14  //DDN    DD  DISP=(,CATLG),UNIT=2400-1,DSN=YEAR.MON(+1),             X
    //           LABEL=(,SUL),DCB=A.B.C,VOLUME=SER=GDG18
```

```
15  //DDO    DD  UNIT=2311,DSNAME=LNG,DISP=(,KEEP),SPACE=(TRK,(12,2))

16  //DDP    DD  UNIT=2314,DSNAME=CLB,DISP=(,CATLG),                          X
    //           SPACE=(1024,(100,25)),LABEL=(,SUL,,EXPDT=70180),            X
    //           VOL=SER=S12148,DCB=(BLKSIZE=240,RECFM=FB,LRECL=60)

17  //DDQ    DD  UNIT=2311,DSNAME=JCD,DISP=(NEW,KEEP),                       X
    //           SPACE=(CYL,(8,1)),DCB=DSORG=DA

18  //DDR    DD  UNIT=2302,DSN=MT12,DISP=(,PASS),                            X
    //           SPACE=(1024,(200,10)),DCB=(DSORG=DA,BLKSIZE=200,            X
    //           KEYLEN=4,RECFM=F),LABEL=(,SUL),VOLUME=SER=49878

19  //DDS    DD  UNIT=2302,DSNAME=PDS14,DISP=(NEW,KEEP),                     X
    //           SUBALLOC=(CYL,(20,1,3),STEP1.DD1)

20  //DDT    DD  UNIT=2314,DSNAME=AHTRY,DISP=(,CATLG),                       X
    //           SPACE=(CYL,(8,2,2)),LABEL=(,,PASSWORD),                     X
    //           VOLUME=SER=158491,DCB=(RECFM=F,LRECL=80)

21  //DDU    DD  DSNAME=AHTRY(SET4),DISP=OLD

22  //DDV    DD  UNIT=2302,VOLUME=SER=X13912,DISP=OLD,                       X
    //           DSNAME=SHTR(MEMB2)

23  //DDW    DD  UNIT=2311,DSNAME=DAT(PRIME),DISP=(NEW,KEEP),                X
    //           DCB=DSORG=IS,SPACE=(CYL,(5,,1))

24  //DDX    DD  UNIT=2302,DSN=ISQ(PRIME),DISP=(,KEEP),DCB=(DSORG=IS,        X
    //           BLKSIZE=240,CYLOFL=1,OPTCD=MYLR,RECFM=FB,LRECL=60,          X
    //           RKP=19,KEYLEN=10),SPACE=(CYL,2),VOL=SER=535861,             X
    //           LABEL=EXPDT=70301
    //       DD  UNIT=2302,DSN=ISQ(OVFLOW),DISP=(,KEEP),DCB=*.DDX,           X
    //           SPACE=(CYL,1),VOL=SER=538267,LABEL=EXPDT=70301

25  //DDY    DD  DSNAME=PAY.WEEK(+1),DISP=(,CATLG),UNIT=2314,                 X
    //           SPACE=(TRK,(3,2))

26  //DDZ    DD  DSN=INV.FORM8(+2),DISP=(,CATLG),UNIT=2311,                   X
    //           VOLUME=SER=SA2103,LABEL=(,SUL),DCB=(MODEL2,RECFM=F,         X
    //           LRECL=80),SPACE=(CYL,(2,1))

27  //DD1    DD  DSNAME=A.B.C,DISP=OLD

28  //DD2    DD  DSN=KELL12,DISP=OLD,LABEL=(,NSL),UNIT=(,P),                  X
    //           DCB=(BUFNO=4,HIARCHY=1)

29  //DD3    DD  DSNAME=FILE18,UNIT=2400,DISP=OLD,VOL=SER=96977

30  //DD4    DD  DSNAME=MILS,UNIT=2400-2,DISP=(OLD,PASS),VOL=SER=9818,       X
    //           LABEL=(,NSL),DCB=(BLKSIZE=1600,LRECL=80)

31  //DD5    DD  DSNAME=GLOSS,DISP=OLD,UNIT=2311,VOLUME=SER=P14992

32  //DD6    DD  DSNAME=LAB14,UNIT=2301,DISP=OLD,VOLUME=SER=H69568,          X
    //           LABEL=(,SUL,,IN)
```

```
33  //DD7    DD  DSNAME=SERNOS,DISP=OLD,UNIT=2311,VOLUME=SER=X20

34  //DD8    DD  DSN=BOLS,DISP=OLD,VOLUME=SER=W5898,UNIT=2302,          X
    //           LABEL=(,SUL)


35  //DD9    DD  DSN=PGM(A81),DISP=OLD

36  //DD10   DD  DSNAME=LIBS(PROJ6),UNIT=2301,DISP=OLD,VOL=SER=D4762


37  //DD1    DD  DSNAME=IND31,UNIT=(2311,2),DISP=OLD,VOLUME=SER(C2021,  X
    //           C2022),DCB=DSORG=IS


38  //DD12   DD  DSNAME=CHAN,DISP=(OLD,KEEP)

39  //DD13   DD  DSNAME=*.STEP1.CREATE,DISP=(OLD,DELETE),LABEL=(,NL),   X
    //           UNIT=(,2),VOLUME=(PRIVATE,,4),DCB=*.STEP1.CREATE


40  //DD14   DD  DSNAME=PAY.WEEK(-3),DISP=OLD

41  //DD15   DD  DSN=INV.FORM8(0),DISP=OLD,LABEL=(,SUL),UNIT=(,P),      X
    //           DCB=(BLKSIZE=240,RECFM=FB,LRECL=60)
```

Section XII: Glossary

* parameter: This parameter is coded as the first parameter on a DD statement that precedes data in the input stream.

ACCT parameter: This parameter is used to supply accounting information for a job step to an installation accounting routine and is coded on an EXEC statement.

AFF parameter: This parameter is used to request the same channel separation from certain data sets as was requested earlier in the job step. The AFF parameter is coded on a DD statement.

alias: An alternate name that may be used to refer to a member of a partitioned data set.

allocation: The process of assigning a resource to a job step.

automatic restart: A restart of a job after a job step abnormally terminates. The restart takes place during the current run, that is, without resubmitting the job.

automatic volume recognition (AVR): A feature that allows the operator to mount labeled volumes on available input/output devices before those volumes are required by a job step.

auxiliary storage: Data storage other than main storage; secondary storage.

background job: A job that is entered through a time sharing terminal by means of the SUBMIT command or through the input stream (SYSIN).

backward reference: A facility of the job control language that permits you to copy information or refer to DD statements that appear earlier in the job.

block prefix: An optional field that may precede the first or only record in a block. For D-format records, the block prefix can contain the actual block length.

catalog:
1. The collection of all data set indexes maintained by data management. Each entry contains a data set name and volume and unit information about the data set.
2. To place an entry for a data set in the catalog. To specify this on a control statement, code DISP=(status,CATLG) on the DD statement that defines the data set you want cataloged. You can retrieve a data set using less information on the DD card if the data set is cataloged.

cataloged data set: A data set that is represented in an index or hierarchy of indexes in the system catalog, the indexes provide the means for locating the data set.

cataloged procedure: A set of job control statements that has been assigned a name and placed in a partitioned data set known as the procedure library. To use a cataloged procedure, code the procedure name on an EXEC statement.

checkpoint/restart: A facility of the operating system that can minimize time lost in reprocessing a job step that abnormally terminated. The CHKPT macro instruction, the RESTART parameter on the JOB statement, and the RD parameter on the JOB or EXEC statement are associated with this facility.

checkpoint restart: A restart within a job step. The restart may be automatic (depending on an eligible completion code and the operator's consent) or deferred, where deferred involves resubmitting the job and coding the RESTART parameter on the JOB statement of the resubmitted job.

CLASS parameter: This parameter is used to assign a job class to your job and is coded on a JOB statement. In multiprogramming systems, jobs within a job class are initiated according to their priority numbers.

command statement: A job statement that is used to issue commands to the system through the input stream.

comment statement: A job control statement used to contain information that may be helpful to yourself or another person that may be running your job or reviewing your output listing.

concatenated data sets: A group of input data sets that are treated as one data set for the duration of a job step.

COND parameter: This parameter is used to test return codes issued by the processing programs; any test that is satisfied causes the job to be terminated or a job step to be bypassed. The COND parameter is coded on a JOB or EXEC statement.

control volume: A volume that contains one or more indexes of the catalog.

data control block (DCB): A control block used to contain certain attributes required by an access method to store or retrieve a data set. The DCB parameter is one means of supplying attributes.

DATA parameter: This parameter is coded as the first parameter on a DD statement that precedes data in the input stream when the data contains job control statements.

data set: An organized collection of related data in one of several prescribed arrangements. The information required to store and retrieve this data is defined on a DD statement.

data set control block: A data set label for a data set on a direct access volume.

data set label: A collection of information that describes the attributes of a data set. The data set label for a data set is normally on the same volume as the data set it describes.

DCB: See data control block.

DCB parameter: This parameter is used to supply attributes about the data set that are needed to complete the data control block. The DCB parameter is coded on a DD statement.

D format: A data set format in which ASCII records are variable lengths.

DD (data definition) statement: A job control statement that defines a data set that is being created or retrieved in a job step. DD statements follow an EXEC statement.

ddname (data definition name): A name assigned to a DD statement. This name corresponds to the ddname appearing in a data control block.

DDNAME parameter: This parameter is used to postpone the definition of a data set until later in the same job step and is coded on a DD statement.

deferred restart: A restart that is performed when a job is resubmitted and the RESTART parameter is coded on the JOB statement of the resubmitted job.

delimiter statement: A job control statement used to mark the end of data. The characters /* appear in columns 1 and 2 of this control statement.

device type: A number that corresponds to a type of input/output device. Coding the device type in the UNIT parameter is one way of indicating what input/output device you want allocated to a job step.

direct access device: An auxiliary storage device in which the access time is effectively independent of the location of the data set.

direct data set: A data set whose records are in random order on a direct access volume. Each record is stored or retrieved according to its actual address or its address relative to the beginning of the data set.

directory: A series of 256-byte records at the beginning of a partitioned data set that contains an entry for each member in the data set.

DISP parameter: This parameter is used to describe the status of the data set and indicates what should be done with the data set after termination of the job step that processes it, or at the end of the job. The DISP parameter is coded on a DD statement.

dispatching priority: The number assigned to a task, which in a multitask environment, determines the order in which the tasks may use main storage and CPU resources.

DLM parameter: This parameter defines a special delimiter to be used to terminate a group of data in the input stream.

DPRTY parameter: This parameter is used to assign a dispatching priority to a job step and is coded on an EXEC statement.

DSN parameter: This parameter is used to assign a name to a new data set or to identify an existing data set and is coded on a DD statement. Coding DSN is the same as coding DSNAME.

DSNAME parameter: This parameter is used to assign a name to a new data set or to identify an existing data set and is coded on a DD statement. Coding DSNAME is the same as coding DSN.

DUMMY parameter: This parameter is used to tell the system that the processing program should be executed, but no input or output operations should be performed on a particular data set. The DUMMY parameter is coded as the first parameter on a DD statement.

DYNAM parameter: For TSO, this parameter is used to specify that dynamic allocation of data sets is to be used. This allows you to defer definition of data set until you require it. If DYNAM is used in the background (batch environment), it means the same as DUMMY. The DYNAM parameter is coded on a DD statement.

dynamic storage: That portion of main storage that is subdivided into partitions or regions for use by the programs associated with job steps and some system tasks.

Exclusive control of a data set: This means that only one job at a time can process a data set. A request for an exclusively controlled data set will not be processed until the job with control terminates. Also a request for the data set name itself will not be processed -- even though the name may not refer to the same physical data set.

EXEC (execute) statement: A job control statement that marks the beginning of a job step and identifies the program to be executed or the cataloged or in-stream procedure to be used.

extent: A contiguous area of storage on a direct access volume in which a data set resides. A data set may reside in more than one area of storage on one or more volumes.

F format: A data set format in which the logical records are the same length.

FCB parameter: This parameter is used to specify the forms control image you want to use to print an output data set on a 3211 printer. The FCB parameter is coded on a DD statement.

fixed-length record: A record having the same length as all other records with which it is logically or physically associated.

foreground: The environment in which programs invoked by commands are performed. Programs are swapped in and out of main storage as necessary to efficiently utilize main storage.

foreground job: Any job executing in a foreground region, such as a command processor or a terminal user's program. Also called a "terminal job."

generation data group: A collection of data sets that are kept in chronological order; each data set is called a generation. The DSNAME parameter is used to define the generation you are creating or retrieving.

generation data set: One generation of a generation data group.

group name: A 1- to 8-character name that identifies a device or a collection of devices. Coding a group name in the UNIT parameter is one way of indicating what type of input/output device you want allocated to a job step.

index:
1. A table in the catalog used to locate data sets.
2. A table used to locate the records of an indexed sequential data set.

indexed sequential data set: A data set on one or more direct access volumes whose records contain a key portion, and the location of each record depends on the contents of the key portion. The location of each record is computed through the use of an index.

initiation: The process of selecting a job step for execution and allocating input/output devices for the job step.

input job queue: A queue of summary information of job control statements maintained by the job scheduler, from which it selects the jobs and job steps to be processed.

input stream: The sequence of control statements and data submitted to the operating system on an input device especially activated for this purpose by the operator.

In-stream procedures: A set of job control statements, beginning with a PROC statement and ending with a PEND statement, that have been placed in the input stream. An in-stream procedure can be executed any number of times during the job in which it appears.

job: A total processing application that consists of one or more processing programs required to perform the application. A job is identified by a JOB statement.

JOB statement: A job control statement that marks the beginning of a job, and when jobs are stacked in the input stream, marks the end of the control statements for the preceding job.

job class: An alphabetic character of A through O that characterizes the type of job you are submitting. Each job class is defined by the installation; you indicate the type of job you are submitting in the CLASS parameter on the JOB statement. In multiprogramming systems, jobs within a job class are initiated according to their priority numbers.

job control language: A high-level programming language used to code job control statements, which describe a job to the operating system and inform the system of how the job is to be processed.

job control statement: Any one of the control statements in the input stream that identifies a job or defines its requirements.

job library: See private library.

job management: A general term that collectively describes the functions of the job scheduler and master scheduler.

job processing: The reading of control statements and data from an input stream, the initiating of job steps defined in these statements, and the writing of system output messages.

job scheduler: A control program function that controls input streams and system output, obtains input/output devices for jobs and job steps, and regulates the use of the computing system by jobs. The job scheduler is made up of the reader/interpreter, initiator/terminator, and output writer.

job step: The unit of work associated with one processing program or one cataloged or in-stream procedure, and related data. A job consists of one or more job steps.

JOBLIB: A special ddname that when specified on a DD statement indicates to the system that you are defining a private library.

jobname: The name assigned to a JOB statement; it identifies the job to the system.

K: 1024 bytes.

keyword: A symbol that identifies a parameter or subparameter.

keyword parameter: A parameter that consists of a keyword followed by an equal sign, followed by a single value or a list of subparameters. Keyword parameters must follow positional parameters in the operand field of a job control statement, but the keyword parameters may appear in any order.

LABEL parameter: This parameter is used: (1) to describe the data set label associated with the data set; (2) to describe the sequence number of a data set that does not reside first on a reel; (3) to assign a retention period; (4) to assign password protection; and (5) to override the OPEN macro instruction (ESAM only). The LABEL parameter is coded on a DD statement.

library:
1. In general, a collection of information associated with a particular use, and the location of which is identified in a directory of some type. In this context, see link library, private library, system library.
2. Any partitioned data set.

limit priority: A priority associated with every task in an MVT system, representing the highest dispatching priority that the task may assign to itself or to any of its subtasks.

link library: A partitioned data set named SYS1.LINKLIB. Each member is a processing program and is called in the PGM parameter on the EXEC statement or in the ATTACH, LINK, LOAD, and XCTL macro instructions.

logical record: A record that is defined in terms of the information it contains rather than by its physical traits. You may have to specify the length of the logical record to complete the data control block; one way to specify this is in the LRECL subparameter of the DCB parameter.

main storage: All addressable storage from which instructions can be executed or from which data can be loaded directly into registers.

main storage hierarchy support: An option that divides main storage into two blocks known as hierarchies; hierarchy 0 is assigned to processor storage and hierarchy 1 to the IBM 2361 Core Storage unit.

master scheduler: The part of the control program that responds to operator commands and returns required information.

member: An independent, sequentially organized data set identified by a unique name in a data set directory.

Message Control Program (MCP): A set of user-defined TCAM routines that identify the teleprocessing network to the IBM System/360 Operating System, establish the line control required for the various kinds of stations and modes of connection, and control the handling and routing of

messages in accordance with the user's requirements.

MFT (multiprogramming with a fixed number of tasks): A control program that provides priority scheduling of a fixed number of tasks. A priority scheduler is used in MFT.

MSGCLASS parameter: This parameter is used to assign an output class to the system messages for your job and is coded on a JOB statement.

MSGLEVEL parameter: This parameter is used to indicate what job control statements and allocation/termination messages you want displayed as output from your job and is coded on a JOB statement.

multiprogramming: Executing more than one job step concurrently.

mutually exclusive: The term applied to two parameters that cannot be coded on the same job control statement.

MVT (multiprogramming with a variable number of tasks): A control program that provides priority scheduling of a variable number of tasks. A priority scheduler is used in MVT.

MVT with Model 65 multiprocessing: An extension of MVT. This control program is used with the Model 65 multiprocessing (M65MP) system.

M65MP: See MVT with Model 65 multiprocessing.

name: A 1- to 8-character term, beginning with an alphabetic or national (#, @, $) character, that identifies a data set, a control statement, a program, or a cataloged or in-stream procedure.

nonspecific volume request: A request for volumes that allows the system to select suitable volumes. This type of request can only be made when defining an output data set.

nontemporary data set: A new data set that exists after the job that created it terminates.

NOTIFY parameter: This parameter indicates to the system that a message is to be sent to your time sharing terminal when your job completes. The NOTIFY parameter is coded on the JOB statement.

null statement: A job control statement used to mark the end of a job's control statements and data.

OUTLIM parameter: This parameter is used to specify the maximum number of logical records you want included for the output data set being routed through the output stream. The OUTLIM parameter is coded on a DD statement that must also contain the SYSOUT parameter.

output class: An alphabetic or numeric character that characterizes the type of output data to be written to a unit record device. Each output class is defined by the installation. For system messages, you indicate the type of output data in the MSGCLASS parameter on a JOB statement; for output data sets, you indicate the type of output data in the SYSOUT parameter on a DD statement.

output listing: A form that is printed at the end of your job that may contain job control statements used by your job, diagnostic messages about your job, data sets created by your job, or a dump.

output stream: Diagnostic messages and other output data issued by the operating system or the processing program on output devices especially activated for this purpose by the operator.

output writer: A part of the job scheduler that writes output data sets onto a system output device, independently of the programs that produced the data sets. It also writes system output messages.

PARM parameter: This parameter is used to supply a processing program with information it requires at the time the program is executed and is coded on an EXEC statement.

parameter: A character string that is recognized as having meaning by the reader/interpreter. For most of these character strings, variable information is provided to give a constant value for a specific process or purpose.

partition: In systems with MFT, a subdivision of the dynamic area of main storage set aside for a job step or a system task.

partitioned data set: A collection of independent groups (called members) of sequential records on a direct access volume. Each member has a unique name and is listed in a directory at the beginning of the data set.

PEND statement: A job statement used to mark the end of an in-stream procedure.

PGM parameter: This parameter appears as the first parameter on an EXEC statement when you want to execute a particular program.

physical record: A record that is defined in terms of physical qualities rather than by the information it contains (logical record).

positional parameter: A parameter that must precede all keyword parameters in the operand field of a job control statement. Positional parameters must appear in a specified order.

primary quantity: The initial amount of space on a direct access volume that you request in the SPACE, SPLIT, or SUBALLOC parameter.

priority: A rank assigned to each job step that determines the order in which job steps are selected for execution and requests for resources are satisfied.

priority scheduler: A scheduler that processes complete jobs according to their initiation priority within job classes. Priority shcedulers can accept input data from more than one input stream.

private: The term applied to a mounted volume that the system cannot allocate to an output data set for which a nonspecific volume request is made. A private volume is demounted after its last use in a job step.

private library: A partitioned data set whose members are load modules not used often enough to warrant their inclusion in the link library. To execute a program that resides on a private library, you must define that library on a DD statement that has the ddname JOBLIB or STEPLIB.

PROC parameter: This parameter appears as the first parameter on an EXEC statement when you want to call a particular cataloged or in-stream procedure.

PROC statement: A job control statement used in cataloged or in-stream procedures. It can be used to assign default values for symbolic parameters contained in a procedure. For in-stream procedures, it is used to mark the beginning of the procedure.

procedure step: That unit of work associated with one processing program and related data within a cataloged or in-stream procedure. A cataloged or in-stream procedure consists of one or more procedure steps.

processing program: Any program capable of operating in the system as a problem program does. This includes IBM-distributed language processors, application programs, service and utility programs, and user-written programs.

PRTY parameter: This parameter is used to indicate the job's initiation priority within its job class and is coded on a JOB statement.

public: The term applied to a mounted volume that the system can allocate to an output data set for which a nonspecific volume request is made. A public volume remains mounted until the device on which it is mounted is required by another volume.

QNAME parameter: This parameter allows the user to access messages received by means of TCAM for processing by an application program. It is coded on the DD statement.

qualified name: A data set name that is composed of multiple names separated by periods (e.g., A.B.C.). For a cataloged data set, each name corresponds to an index level in the catalog.

RD parameter: This parameter is used to define the type of restart that can occur and is coded on a JOB or EXEC statement.

reader/interpreter: A job scheduler function that analyzes an input stream of job control statements.

record: A general term for any unit of data that is distinct from all others.

region: In systems with MVT, a subdivision of the dynamic area of main storage set aside for a job step or a system task. You can specify in the REGION parameter on the JOB statement or EXEC statement how large this area of main storage should be.

REGION parameter: This parameter is used to specify how much contiguous main storage is required to execute a job step and can be coded on a JOB or EXEC statement. If main storage hierarchy support is included in the system, the REGION parameter is also used to identify the hierarchy or hierarchies in which the storage is to be allocated.

resource: Any facility of the computing system or operating system required by a job or task and includes main storage, input/output devices, the CPU, data sets, and control and processing programs.

restart: The process of resuming a job after it abnormally terminates. When a restart is performed, processing is continued either at the beginning of a job step that caused the abnormal termination or at a checkpoint within this job step.

RESTART parameter: This parameter is used to identify the step or the step and the checkpoint within the step at which execution of a job is to be resumed and is coded on the JOB statement of a resubmitted job that is to use the checkpoint/restart facilities.

ROLL parameter: This parameter is used to specify a job step's ability to be rolled out or to cause rollout of another job step and is coded on a JOB or EXEC statement.

rollout/rollin: An optional MVT control program feature that allows the temporary assignment of additional main storage to a job step.

scheduler: See job scheduler.

secondary quantity: The additional amount of space on a direct access volume that you want allocated to a data set if the primary quantity requested in the SPACE, SPLIT, or SUBALLOC parameter is not sufficient.

secondary storage: See auxiliary storage.

SEP parameter: This parameter is used to request channel separation from specific data sets defined earlier in the job step and can be coded on a DD statement.

sequential data set: A data set whose records are organized on the basis of their successive physical positions, such as they are on magnetic tape.

Shared control: This means that jobs that are executing simultaneously with a job step that specifies SHR for a data set can use that data set if they also specify SHR for that data set name.

SPACE parameter: This parameter is used to indicate how much space should be allocated on a direct access volume for a new data set and is coded on a DD statement.

specific volume request: A request for volumes that informs the system of the volume serial numbers.

SPLIT parameter: This parameter is used to allocate space to two or more new data sets that are to share cylinders. The SPLIT parameter is coded on a DD statement.

station: In TCAM, either a remote terminal, or a remote computer used as a terminal.

STEPLIB: A special ddname that when specified on a DD statement indicates to the system that you are defining a private library.

stepname: The name assigned to an EXEC statement; it identifies a job step within a job.

step restart: A restart at the beginning of a job step that abnormally terminates. The restart may be automatic (depending on an eligible completion code and the operator's consent) or deferred, where deferred involves resubmitting the job and coding the RESTART parameter on the JOB statement of the resubmitted job.

storage volume: The main function of a storage volume is to contain nontemporary data sets for which a nonspecific volume request was made and PRIVATE was not coded in the VOLUME parameter. A direct access volume becomes a storage volume when so indicated in a MOUNT command or in a member of SYS1.PARMLIB named PRESRES.

SUBALLOC parameter: This parameter is used to place a series of a new data sets in one area of contiguous space on a direct access volume and in a certain sequence. The SUBALLOC parameter is coded on a DD statement.

subparameter: One of the items of variable information that follows a keyword parameter and can be either positional or keyword.

symbol: In the IBM System/360 Operating System, any group of eight or less alphameric and national characters that begins with an alphabetic or national (#,@,$) character.

symbolic parameter: A symbol preceded by an ampersand that appears in a cataloged procedure. Values are assigned to symbolic parameters when the procedure in which they appear is called.

SYSABEND: A special ddname that when specified on a DD statement tells the system you are defining a data set on which a dump can be written if the step abnormally terminates. The dump provided includes the system nucleus, the processing program storage area, and possibly a trace table.

SYSCHK: A special ddname that when specified on a DD statement that precedes the first EXEC statement in the job tells the system you are defining a data set that contains checkpoint entries. This DD statement is included in a job that is being resubmitted for execution and execution is to begin at a particular checkpoint.

SYSCTLG: The name of a system data set that contains the name and location of cataloged data sets.

SYSIN: A name conventionally used as the data definition name of a data set in the input stream.

SYSOUT parameter: This parameter is used to assign an output class to an output data set and can be coded on a DD statement.

system data sets: The data sets that make up the IBM System/360 Operating System.

system generation: The process of producing an operating system made up of standard and optional components.

system input device: A device specified as a source of an input stream.

system library: One of the collection of all cataloged data sets at an installation.

system management facilities: An optional control program feature that provides the means of gathering and recording information that can be used to evaluate system usage.

system messages: Messages issued by the system that pertain to a problem program. These messages appear on an output listing and may include such messages as error messages, disposition messages, and allocation/de-allocation messages.

system output device: An output device, shared by all jobs, onto which specified output data is written.

SYSUDUMP: A special ddname that when specified on a DD statement tells the system you are defining a data set on which a dump can be written if the step abnormally terminates. The dump provided is the processing program storage area.

SYS1.LINKLIB: The name of a partitioned data set that contains IBM-supplied processing programs and part of the nonresident portion of the control program. It may also contain user-written programs.

SYS1.PROCLIB: The name of a partitioned system data set that contains cataloged procedures.

SYS1.SYSJOBQE: A system data set that contains information about the input and output streams, and contains the input and output queues.

task: The smallest unit of work that can be performed under the control program.

Telecommunications Access Method (TCAM): The combination of an access technique and a given data set organization in a teleprocessing application that allows the programmer to transfer data between main storage and remote I/O devices.

temporary data set: A new data set that is created and deleted in the same job.

TERM parameter: This parameter is used to indicate to the system that the input or output data being defined is coming from or going to a time sharing terminal.

terminal table: An ordered collection of information consisting of a control field for the table and blocks of information on each line, station, component, or application program from which a message can originate or to which a message can be sent.

termination: The process of performing disposition processing, as specified in the DISP parameter, de-allocating input/output devices, and supplying control information for writing job output on a system output unit.

TIME parameter: This parameter is used to assign a time limit on how long the job or a particular job step can use the CPU and is coded on a JOB or EXEC statement, or both.

time sharing: A method of using a computing system that allows a number of users to execute programs concurrently and to interact with the programs during execution.

Time Sharing Option (TSO): An option of the operating system providing conversational time sharing from remote terminals.

time-slicing: The sharing of the CPU by certain tasks for an equal, predetermined length of time.

TYPRUN parameter: This parameter is used to hold a job for execution until the operator issues a RELEASE command and is coded as TYPRUN=HOLD or a JOE statement.

UCS parameter: This parameter is used to describe the character set you want to use for printing an output data set on a 1403 printer. The UCS parameter is coded on a DD statement.

unit address: A 3-byte number, made up of the channel, control unit, and unit numbers, that identifies a particular device. Coding a unit address in the UNIT parameter is one way of indicating what input/output device you want allocated to the job step.

UNIT parameter: This parameter is used to describe what device and how many devices you want assigned to a data set. The UNIT parameter can be coded on a DD statement.

V format: A data set format in which logical records are of varying length and include a length indicator; and in which V format logical records may be blocked, with each block containing a block length indicator.

VOL parameter: This parameter is used to identify the volume(s) on which a data set resides or will reside and is coded on a DD statement. Coding VOL is the same as coding VOLUME.

volume: That portion of an auxiliary storage device that is accessible to a single read/write mechanism.

VOLUME parameter: This parameter is used to identify the volume(s) on which a data set resides or will reside and is coded on a DD statement. Coding VOLUME is the same as coding VOL.

volume table of contents (VTOC): A table in a direct access volume that describes each data set on the volume.

Index

Indexes to systems reference library manuals are consolidated in the publication IBM System/360 Operating System: Systems Reference Library Master Index, C28-6644. For additional information about any subject listed below, refer to other publications listed for the same subject in the Master Index.

Where more than one page reference is given, the major reference is first.

Index

The JOB Statement

//Name	Operation	Operand	P/K	Comments
//jobname	JOB	([account number][,additional accounting information,...])	P	Can be made mandatory
		[programmer's name]	P	Can be made mandatory
		[CLASS=jobclass]	K	Assign A-O.
		[COND=((code,operator),...)]	K	Maximum of 8 tests
		[MSGCLASS=output class]	K	Assign A-Z,0-9.
		[MSGLEVEL=([0|1|2][,0|,1])]	K	
		[NOTIFY=user identification]	K	Notify user of job completion. For MVT with TSO.
		[PRTY=priority]	K	Assign 0-13.
		[RD={R|RNC|NC|NR}]	K	Restart definition
		[REGION=({valueK|value0K}[,value1K])]	K	For MVT
		[RESTART=({*|stepname|stepname.procstepname}[,checkid])]	K	For deferred restart
		[ROLL=({YES|NO}{,YES|,NO})]	K	Rollout/rollin. For MVT.
		[TIME={(minutes,seconds)|1440}]	K	Assigns job CPU time limit.
		[TYPRUN=HOLD]	K	Holding a job in job queue.

Legend:

P Positional parameter.
K Keyword parameter.
{ } Choose one.
[ ] Optional; if more than one line is enclosed, choose one or none.

Chart 1

Figure 43. Job Statement Chart

The EXEC Statement

//Name	Operation	Operand	P/K	Comments
//[stepname]	EXEC	{PGM={program name / *.stepname.ddname / *.stepname.procstepname.ddname} / [PROC=]procedure name}	P	Identifies program or cataloged procedure
		[ACCT=(accounting information,...) / ACCT.procstepname=(accounting information,...)]	K	Accounting information for step
		[COND=([(code,operator) / (code,operator,stepname) / (code,operator,stepname.procstepname)],...[,][EVEN / ONLY]) / COND.procstepname=([(code,operator) / (code,operator,stepname) / (code,operator,stepname.procstepname)],...[,][EVEN / ONLY])]	K	Maximum of 8 tests, or 7 tests if EVEN or ONLY is coded
		[DPRTY=(value1,value2) / DPRTY.procstepname=(value1,value2)]	K	Assign values of 0-15. For MVT.
		[PARM=value / PARM.procstepname=value]	K	Parentheses or apostrophes enclosing value may be required
		[RD={R / RNC / NC / NR} / RD.procstepname={R / RNC / NC / NR}]	K	Restart definition
		[REGION=({valueK / value0K}[,value1K]) / REGION.procstepname=({valueK / value0K}[,value1K])]	K	For MVT
		[ROLL=({YES / NO}{,YES / ,NO}) / ROLL.procstepname=({YES / NO}{,YES / ,NO})]	K	Rollout/rollin. For MVT.
		[TIME={(minutes,seconds) / 1440} / TIME.procstepname={(minutes,seconds) / 1440}]	K	Assigns step CPU time limit.

Legend:

P Positional parameter.
K Keyword parameter.
{} Choose one.
[] Optional; if more than one line is enclosed, choose one or none.

Chart 2

Figure 44. Execute Statement Chart

The DD Statement

//Name	Operation	Operand	P/K	Comments
// [ddname / procstepname.ddname]	DD	[* / DATA [,DLM=xx]]	P	To define a data set in the input stream.
		[DUMMY]	P	To bypass I/O operations on a data set (BSAM and QSAM)
		[DYNAM]	P	To request dynamic allocation. For MVT with TSO.
		[AFF=ddname]	K	One way to request channel separation.
		[DCB=(list of attributes) / DCB=(dsname / *.ddname / *.stepname.ddname / *.stepname.procstepname.ddname [,list of attributes])]	K	To complete the data control block. See Glossary of DCB Subparameters.
		[DDNAME=ddname]	K	To postpone the definition of a data set.
		[DISP=([NEW / OLD / SHR / MOD] [,DELETE / ,KEEP / ,PASS / ,CATLG / ,UNCATLG / ,] [,DELETE / ,KEEP / ,CATLG / ,UNCATLG])]	K	To assign a status, disposition, and conditional disposition to the data set
		[DLM=delimiter]	K	Assigns a delimiter other than /*
		[{DSNAME / DSN}=dsname / dsname(member name) / dsname(generation number) / dsname(area name) / &&dsname / &&dsname(member name) / &&dsname(area name) / *.ddname / *.stepname.ddname / *.stepname.procstepname.ddname]	K	To assign a name to a new data set or to identify an existing data set. An unqualified name is 1-8 characters, beginning with an alphabetic or national character.
		[FCB=(image-id [,ALIGN / ,VERIFY])]	K	To specify forms control information. The FCB parameter is ignored if the data set is not written to a 3211 printer.
		[LABEL=([data set seq #] [,SL / ,SUL / ,AL / ,AUL / ,NSL / ,NL / ,BLP / ,LTM / ,] [,PASSWORD / ,NOPWREAD] [,IN / ,OUT] [,] [EXPDT=yyddd / RETPD=nnnn])]	K	To supply label information
		[OUTLIM=number]	K	To limit the number of logical records you want included in the output data set.
		[QNAME=process name]	K	Specifies the name of a TPROCESS macro which defines a destination queue for messages received by means of TCAM.

Chart 3

Figure 45. Data Definition Statement Chart

The DD Statement (con't)

//Name	Operation	Operand	P/K	Comments
		[{ [1] SPACE=({ TRK / CYL / blocklength } ,(primary [,secondary] [,directory] [,index]) [,RLSE] [, / ,CONTIG / ,MXIG / ,ALX] [,ROUND]) / [2] SPACE=(ABSTR,(primary quantity,address [,directory] [,index])) }]	K	[1] To assign space on a direct access volume for a new data set [2] To assign specific tracks on a direct access volume for a new data set
// [ddname / procstepname.ddname]	DD	[SPLIT= { (n,CYL,(primary quantity [,secondary quantity])) / n,(percent,blocklength,(primary quantity [,secondary quantity])) / percent }]	K	To assign space on a direct access volume for a new data set. Data sets share cylinders.
		[SUBALLOC=({ TRK / CYL / blocklength } ,(primary [,secondary] [,directory]) { ,ddname / ,stepname.ddname / ,stepname.procstepname.ddname })]	K	To request part of the space on a direct access volume assigned earlier in the job
		[SYSOUT=(classname [,program name] [,form number]), [OUTLIM=number]]	K	To route a data set through the output stream. For classname, assign A-Z or 0-9.
		[TERM=TS]	K	To indicate to the system that the input or output data being defined is coming from or going to a time sharing terminal.
		[UCS=(character set code [,FOLD] [,VERIFY])]	K	To request a special character set for a 1403 printer
		[{ UNIT=([unit address / device type / group name] [,unit count / ,P / ,] [,DEFER] [,SEP=(ddname,...)]) / UNIT=AFF=ddname }]	K	To provide the system with unit information
		[{ VOLUME / VOL }=([PRIVATE] [,RETAIN] [, / ,volume seq #] [,volume count] [,] [SER=(serial number,...) / REF=dsname / REF=*.ddname / REF=*.stepname.ddname / REF=*.stepname.procstepname.ddname])]	K	To provide the system with volume information

Legend:

P Positional parameter.

K Keyword parameter.

{ } Choose one.

[ ] Enclosing subparameter, indicates that subparameter is optional; if more than one line is enclosed, choose one or none.

[ ] Enclosing entire parameter, indicates that parameter may be optional, depending on what type of data set you are defining.

IBM System/360 Operating System:
JCL Reference

GC28-6704-4

READER'S COMMENT FORM

Your views about this publication may help improve its usefulness; this form will be sent to the author's department for appropriate action. Using this form to request system assistance or additional publications will delay response, however. *For more direct handling of such requests, please contact your IBM representative or the IBM Branch Office serving your locality.*

Possible topics for comment are:

Clarity Accuracy Completeness Organization Index Figures Examples Legibility

What is your occupation? _____
Number of latest Technical Newsletter (if any) concerning this publication: _____
Please indicate in the space below if you wish a reply.

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. Elsewhere, an IBM office or representative will be happy to forward your comments.

Cut or Fold Along Line

Cut or Fold Along Line

Your comments, please . . .

This manual is part of a library that serves as a reference source for system analysts, programmers, and operators of IBM systems. Your comments on the other side of this form will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Fold Fold

First Class
Permit 81
Poughkeepsie
New York

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

Postage will be paid by:

International Business Machines Corporation
Department D58, Building 706-2
PO Box 390
Poughkeepsie, New York 12602

Fold Fold

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)

System/360 OS JCL Reference (S360-36) Printed in U.S.A. GC28-6704-4