

GC24-5099-4
File No. S370-36

Systems

OS/VS1 JCL Reference

Releases 6.7 and 7



Fifth Edition (January 1979)

This edition, GC24-5099-4, with Technical Newsletter GN24-5628 applies to Releases 6.7 and 7 of OS/VS1 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 *IBM System/370 Bibliography*, GC20-0001 for the editions that are applicable and current.

This edition, GC24-5099-4, is a major revision of GC24-5099-3 and incorporates:
GN24-5533 (a TNL; December 15, 1976)
GN24-5153 (SU5, OS/VS1 MSS Enhancements: March 15, 1977).

Changes and additions to the text and illustrations are indicated by a vertical line to the left of the change.

Summary of Amendments

For a list of changes, see page 3.

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

This publication has been produced by IBM Corporation, Programming Publications, Dept. G60, PO Box 6, Endicott, New York, U.S.A. 13760.

Summary of Amendments for GC24-5099-4 as updated by GN24-5628 VS1 Releases 6.7 and 7

This Technical Newsletter supports those OS/VS1 Release 6.7 items listed previously as well as Mass Storage System Extensions, Program Number 5740-XYG.

For OS/VS1 Release 7, the 3278-2A Display Console is supported as a valid device type.

Summary of Amendments for GC24-5099-4 VS1 Release 6.7

This revision reflects the availability of OS/VS1 Release 6.7. It contains information for:

- IBM 3895 Document Reader/Inscriber (SU4)
- MSS Enhancements (SU5)
- Subsystem Attachment Support (SU6)

- IBM 3033, 3032, 3031 Processor Support (SU20)

Miscellaneous

Miscellaneous changes have also been made to improve the accuracy of the manual.

Summary of Amendments for GC24-5099-3 VS1 Release 6

This revision reflects the availability of OS/VS1 Release 6. Included is:

- IBM System/370 Models 138 and 148 support.
- IBM 3203 Printer support.
- The new parameter COMPACT for the IBM 3790.

- Miscellaneous editorial changes.

Information on the IBM 3138 Display Console Keyboard, 3148 Display Console Keyboard, and 3203 Printer is included for planning purposes only until the products become available.

Preface

This publication defines the facilities provided with the JCL (job control language) and contains the information necessary to code JCL statements. It is intended for use as a reference book by programmers who understand the concepts of job management and data management.

This publication contains:

1. Programming notes (Section I) — Describes JCL coding conventions.
2. Statement Descriptions (Sections II - X) — Each of these sections defines a statement, lists related publications for details, lists coding rules (with at least one example), and devotes a chapter to each parameter.
3. Appendixes — Gives additional information on JCL facilities including several tables for quick reference.
4. Format Charts — Charts of JOB, EXEC, and DD statement parameters follow the appendixes.

The following OS JCL parameters do not apply to OS/VS1. If left unchanged on JCL statements, they are checked for correct syntax by the interpreter, but otherwise are ignored by the system.

ROLL — on the JOB statement and EXEC statement.

HIARCHY — DCB subparameter on the DD statement.

REGION — on the JOB and EXEC statements has been redefined for use with virtual storage.

Before you use this publication, you must understand the concepts and terminology introduced in the prerequisite publication, *OS/VS1 JCL Services*, GC24-5100.

The text also refers to these publications:

OS/VS1 Checkpoint/Restart, GC26-3876.

IBM 2821 Control Unit, GA24-3312.

OS/VS1 Data Management for System Programmers, GC26-3837.

OS/VS1 Data Management Services Guide, GC26-3874.

OS/VS1 Debugging Guide, GC24-5093.

OS/VS1 IBM 3540 Programmer's Reference, GC24-5110.

OS/VS Graphic Programming Services (GPS) for IBM 2250 Display Unit, GC27-6971.

OS/VS Mass Storage System Extensions Services: Guide, SH35-0035.

OS/VS Mass Storage System (MSS) Planning Guide, GC35-0011.

OS/VS1 Planning and Use Guide, GC24-5090.

IBM 3800 Printing Subsystem Programmer's Guide, GC26-3846.

OS/VS1 RES System Programmers's Guide, GC28-6878.

OS/VS1 RES Workstation User's Guide, GC28-6879.

OS/VS1 Supervisor Services and Macro Instructions, GC24-5103.

OS/VS1 System Management Facilities (SMF), GC24-5115.

OS/VS Tape Labels, GC26-3795.

OS/VS Utilities, GC35-0005.

OS/VS Virtual Storage Access Method (VSAM) Programmer's Guide, GC26-3838.

Operator's Library: OS/VS1 CRJE, GC38-0335.

Operator's Library: OS/VS1 Reference, GC38-0110.

For a list of additional publications, refer to the *IBM System/370 Bibliography*, GC20-0001.

Contents

Section I: Programming Notes	11
Fields in Control Statement	12
Parameters in Operand Field	13
Continuing Control Statements	13
Backward References	14
Concatenating Data Sets	15
Character Sets	15
Special Characters	16
Rule for Using Special Characters	16
Section II: The JOB Statement	17
Accounting Information Parameter	19
Programmer's Name Parameter	21
ADDRSPC Parameter	22
CLASS Parameter	23
COND Parameter	24
MPROFILE Parameter	26
MSGCLASS Parameter	27
MSGLEVEL Parameter	28
PROFILE Parameter	30
PRTY Parameter	31
RD Parameter	32
REGION Parameter	34
RESTART Parameter	35
TIME Parameter	37
TYPRUN Parameter	39
Section III: The EXEC Statement	40
PGM Parameter	42
PROC Parameter	45
ACCT Parameter	46
ADDRSPC Parameter	48
COND Parameter	49
PARM Parameter	52
RD Parameter	54
REGION Parameter	56
TIME Parameter	57
Section IV: The DD Statement	59
JOB CAT Facility (VSAM only)	62
JOBLIB Facility	63
STEP CAT Facility (VSAM only)	66
STEPLIB Facility	67
SYSABEND and SYSUDUMP Facilities	70
SYSCHK Facility	72
* Parameter	74
DATA Parameter	76
DUMMY Parameter	78
AFF Parameter	80
AMP Parameter (VSAM only)	82
BURST Parameter	86
CHARS Parameter	87
CHKPT Parameter	89
COMPACT Parameter	91
COPIES Parameter	92
DCB Parameter	94
DCB Subparameters for BDAM	98
DCB Subparameters for BISAM	102
DCB Subparameters for BPAM	104
DCB Subparameters for BSAM	108
DCB Subparameters for BTAM	119
DCB Subparameters for EXCP	121
DCB Subparameters for GAM	127
DCB Subparameters for QISAM	128
DCB Subparameters for QSAM	133
DCB Subparameters for TCAM	144

DDNAME Parameter	149
DEST Parameter (for RES)	151
DISP Parameter	152
DLM Parameter	155
DSID Parameter	157
DSNAME Parameter	158
FCB Parameter	161
FLASH Parameter	163
HOLD Parameter	164
LABEL Parameter	165
MODIFY Parameter	169
MSVGP Parameter	170
OUTLIM Parameter	172
QNAME Parameter	173
SEP Parameter	174
SPACE Parameter	176
SPLIT Parameter	181
SUBALLOC Parameter	184
SUBSYS Parameter	186
SYSOUT Parameter	187
TERM Parameter (for RES)	189
UCS Parameter	190
UNIT Parameter	192
VOLUME Parameter	196
Section V: The Command Statement	202
Section VI: The Comment Statement	205
Section VII: The Delimiter Statement	206
Section VIII: The Null Statement	207
Section IX: The PEND Statement	208
Section X: The PROC Statement	209
Appendix A: Identifying the Data Set to the System	211
Specifying the DDNAME Parameter	211
When You Code the DDNAME Parameter	211
DCB Subparameter BLKSIZE and BUFNO	212
Specifying the DSNAME Parameter	213
Creating or Retrieving a Nontemporary Data Set	213
Nontemporary Data Sets	213
Members of a Partitioned Data Set	214
A Generation Data Group	214
Areas of an Indexed Sequential Data Set	214
Creating or Retrieving a Temporary Data Set	215
Temporary Data Sets	215
Members of a Temporary Partitioned Data Set	215
Areas of a Temporary Indexed Sequential Data Set	216
Using a Dedicated Data Set	216
Identifying Associated Data sets	216
Copying the Data Set Name from an Earlier DD Statement	216
Specifying the DSNAME Parameter in Apostrophes	216
Specifying the LABEL Parameter	217
Data Set Sequence Number Subparameter	218
Label Type Subparameter	218
PASSWORD and NOPWREAD Subparameters	220
IN and OUT Subparameters	220
RETPD and EXPDT Subparameters	220
Appendix B: Reference Tables	222
Retrieving or Extending an Indexed Sequential Data Set	227
Appendix C: Device Types	233
Format Charts	236
Index	241

Figures

Figure 1.	Control Statement Fields	12
Figure 2.	Character Sets	16
Figure 3.	Parameters Containing Special Characters	16
Figure 4.	DD Parameters for Creating a Data Set	223
Figure 5.	DD Parameters for Returning a Data Set	225
Figure 6.	DD Parameters for Extending a Data Set	226
Figure 7.	DD Parameters for Retrieving or Extending an Indexed Sequential Data Set	227
Figure 8.	Area Arrangement of Indexed Sequential Data Sets	228
Figure 9.	Table of Mutually Exclusive DD Parameters	229
Figure 10.	Disposition Processing Chart	230
Figure 11.	Direct Access Capacities	231
Figure 12.	Track Capacities	231
Figure 13.	Character Arrangement Tables Supplied with the 3800	232
Figure 14.	The JOB Statement	236
Figure 15.	The EXEC Statement	237
Figure 16.	The DD Statement	238

Section I: Programming Notes

Notation for Defining Control Statement Parameters

The formats of the parameters for the JOB, EXEC (execute), and DD (data definition) statements appear at the beginning of the chapter on the corresponding parameter. Notations used in these format descriptions are described below.

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

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

2. Lowercase letters, words, and symbols (other than those listed in item 1) — represent variables for which specific information is substituted when the parameter is coded.

Example: PRTY=priority

When the PRTY parameter is coded, 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 and indicate that you must code one of the items.

Example: { TRK
CYL
blocksize }

When this parameter is coded, you must code either TRK, CYL, or a substitute for “blocksize”, 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.

Example: [EXPDT=yyddd
RETPD=nnnn]

When you code this parameter, you can code either EXPDT=yyddd or RETPD=nnnn or omit both.

Sometimes a comma is one of the items enclosed in brackets. You code the comma when no other item in the brackets is used and a following part of the parameter is still to be coded.

Example: [,programe] [,form number]

When you code this parameter, you have the option of coding both “,progrname” and “,form number”, omitting both, or coding only one. If you choose to code only the latter, you must code the first comma as well:

, , form number

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

Example: COND=((code,operator), . . .)

The ellipsis indicates that (code,operator) can be repeated.

Fields in Control Statements

Every control statement is logically divided into as many as four fields:

- name field — identifies the control statement so that other statements and system control blocks can refer to it. The name field is one to eight alphanumeric and national (#, @, \$) characters; the first character must be alphabetical or national. The name field must begin in column 3.
- 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.
- 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 section, “Parameters in the Operand Field”.
- comments field — contains any information deemed helpful by the person who codes the control statement. The comments field must follow the operand field (except on the delimiter statement) and must be preceded by at least one blank.

Figure 1 shows how the fields apply to each statement.

Statement	Columns 1 and 2	Field
job	//	name operation (JOB) operand ¹ comments ²
execute	//	name ¹ operation (EXEC) operand comments ¹
data definition	//	name ¹ operation (DD) operand comments ¹
procedure (Cataloged)	//	name ¹ operation (PROC) operand comments ¹
procedure (in-stream)	//	name operation (PROC) operand ¹ comments ²
procedure end	//	name ¹ operation (PEND) comments ²
command	//	operation (command) operand comments ¹
delimiter	/*	comments ¹
null	//	
Statement	Columns 1, 2, 3	Field
Comments	/*	comments
¹ Optional ² Optional - If operand(s) are not coded, comments cannot be coded. If operand(s) are coded, comments are optional.		

Figure 1. Control Statement Fields

Control statement fields can be coded in free form, except for the name field, which must begin in column 3. Free form coding means that the fields need not begin in a particular column. Insert a blank between fields as a delimiter.

Fields cannot be coded past column 71, except for the comments field, which can be coded through column 80. If the total length of the fields exceeds 71 columns, you must continue the fields on one or more succeeding statements. How to continue fields is described under "Continuing Control Statements".

Note: Columns 73-80 may be used for a sequence number.

Parameters in the Operand Field

The operand field contains two types of parameters:

- positional parameter — characterized by its position in the operand field in relation to other parameters. It must be coded first and in a specific order. For each absent positional parameter, code a comma in its place if followed by another positional parameter. Otherwise, you need not code the comma(s).
- keyword parameter — characterized by a keyword followed by an equal sign and variable information. It is positionally independent of others of its type. Therefore you need not indicate the absence of a keyword parameter. The keyword parameter follows the positional 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. All positional and keyword parameters and subparameters coded in the operand field must be separated by commas.

The EXEC statements and DD statements in cataloged procedures can contain one other type of parameter -- a symbolic parameter. A symbolic parameter, containing a name preceded by an ampersand (&), is a symbol for a parameter, a subparameter, or a value. With symbolic parameters you can vary the information in the operand field of a procedure EXEC or DD statement. The value to be assigned to the symbolic parameter may be coded on the EXEC statement that calls the procedure. This value is in effect only while the procedure is executed. For details on using symbolic parameters in a set of control statements that you plan to catalog as a procedure, see *OS/VSI JCL Services*, GC24-5100.

Continuing Control Statements

When the total length of the fields on a control statement will exceed 71 columns, you must continue the fields on 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 by following the continuation conventions.

To continue the operand field:

1. Interrupt the field after a complete parameter or subparameter, including the comma that follows it, before column 72.
2. To include comments insert at least one blank after the interrupted field.
3. Optionally, code any non-blank character in column 72. (The non-blank 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.
4. Code the identifying characters // in columns 1 and 2 of the following statement.
5. Continue the interrupted operand beginning in any column from 4 through 16.

To continue the comments field:

1. Interrupt the comment at a convenient place before column 72.
2. Code a non-blank character in column 72.
3. Code the identifying characters // in columns 1 and 2 of the following statement.
4. Continue the comments field beginning in any column after column 3.

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 a comment statement, *** appears in columns 1 through 3 on an output listing.

Backward References

You can refer the system to an earlier DD statement in the job for certain information by using the JCL backward reference, which is in this form:

- parameter=*.ddname — use this form when the earlier DD statement is in the same job step.
- parameter=*.stepname.ddname — use this form when the earlier DD statement is in an earlier job step.
- parameter=*.stepname.procstepname.ddname — use this form when the earlier DD statement is 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 from a backward reference are:

- PGM – the data set that contains the program to be executed in this job step.
- DCB – all DCB (data control block) 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

As many as 255 sequential or 16 partitioned input data sets, each of which may reside on a different volume, can be logically connected for the duration of a job step by simply omitting the ddnames from all the DD statements except the first in the sequence. When this ddname is encountered in a DCB 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, (for example, the device types, block lengths, or record formats differ), the DCBOFLGS field of the DCB must be modified while the program is executing. For details, refer to *OS/VS1 Data Management Services Guide*, GC26-3875.

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

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

Do 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=2314,
//           DD      VOLUME=SER=P12,DISP=(OLD,DELETE)
//           DD      *
//           .
//           .
//           .
//           data
//           .
//           .
//           .
/*
```

Character Sets

JCL 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 2.

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	, . / ' () * ε + - = b

Figure 2. Character Sets

Special Characters

Special characters are used in JCL to:

1. Delimit parameters (the comma).
2. Delimit fields (the blank).
3. Perform syntactical functions. (For example, the appearance of εε as the first two characters following DSNNAME= tells the system that a temporary data set name follows.

Sometimes you can code a special character that does not satisfy one of the three uses above. In most of these cases, you must enclose the item that contains the special character in apostrophes (for example, ACCT='123+456'). If one of the special characters is an apostrophe, you must code two consecutive apostrophes in its place, for example, 'O''NEILL'.

Rule for Using Special Characters

Figure 3 contains the parameters that can have special characters as part of their variable information, and indicates when the apostrophes are not required.

Statement	Parameter That Can Have Special Characters As Variable Information	Special Character Not Requiring Apostrophes
JOB	accounting information	hyphens
JOB	programmer's name	periods
JOB	RESTART (checkid field)	none
EXEC	ACCT	hyphens
EXEC	PARM	none
DD	VOLUME (volume serial number)	hyphens
DD	DSNAME	hyphens
DD	DSNAME (qualified name)	periods
DD	DSNAME (temporary data set)	ampersands (as first two characters)
DD	DSNAME (partitioned data set)	parentheses (that enclose name)
DD	DSNAME (generation data group)	parentheses (that enclose name) plus or minus (in generation number)
DD	DSNAME (indexed sequential data set)	parentheses (that enclose name)

Figure 3. Parameters Containing Special Characters.

Section II: The JOB Statement

Control 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.

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

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

Rules for Coding

1. Code a JOB statement for each job. Code a unique jobname in every JOB statement. The jobname must consist of one through eight alphameric and national (#, @, \$) characters; the first character must be alphabetic or national.
2. Two types of parameters can be coded on the JOB statement: positional and keyword.

Positional parameters 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.

You can code keyword parameters in any order after the positional parameters. Any of the following keyword parameters can be coded on the JOB statement:

```
ADDRSPC  
CLASS  
COND  
MPROFILE  
MSGCLASS  
MSGLEVEL  
PROFILE  
PRTY  
RD  
REGION  
RESTART  
TIME  
TYPRUN
```

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

3. All parameters in the operand field are optional unless the account number and programmer's name parameters are required by the installation.
4. If you code no parameters in the operand field of the JOB statement, code no comments.

5. The following names and characters are keywords of the DISPLAY command. Do not use them as jobnames unless you enclose the name used in parentheses.

A
CONSOLES
N
P
PFK
Q
R
RT
SQA
T
TP
U
USER

Sample JOB Statements

//ALPHA	JOB	843,LINLEE,CLASS=F,MSGLEVEL=(1,1)
//LOS	JOB	,BROWNLY,TIME=(4,30),MSGLEVEL=(2,0)
//MART	JOB	1863,RESTART=STEP4
//TRY8	JOB	

Accounting Information Parameter

Positional, Optional (according to installation procedures)

The accounting information parameter identifies an account number and/or any information that the installation has established as a requirement in a PARM field parameter of the cataloged procedure for the input reader.

For information on how to add accounting facilities, refer to *OS/VS1 Planning and Use Guide*, GC24-5090.

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

Rules for Coding

1. Separate the account number and each item of additional accounting information by a comma.
2. When accounting information consists of more than one item, enclose the information in either parentheses or apostrophes, for example '5438,GROUP6' or (5438,GROUP6). If you use apostrophes, all accounting information enclosed in the apostrophes is considered as one field.
3. The account number and other accounting information must not exceed 142 characters in total, including the commas that separate the items.
4. If you must continue the accounting information on another statement, enclose the information in parentheses.
5. If any of the included items contains special characters (except hyphens), either:
 - a. Enclose the accounting information in apostrophes, for example, '5438,10/08/66' or
 - b. Enclose the item in apostrophes and the accounting information in parentheses, for example, (5438,'10/08/66'). The enclosing apostrophes are not considered part of the information.

If one of the special characters is an ampersand and you are not defining a symbolic parameter, code two consecutive ampersands in its place, for example, '34εε8241'.

Examples of the Accounting Information Parameter

//JOB43 JOB D548-868

Account number only; no parentheses are required.

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

Account number plus additional accounting information; parentheses are required.

//JOB45 JOB (, E1659, GROUP6X)

Additional accounting information only; parentheses are required.

Programmer's Name Parameter

Positional, Optional (according to installation procedures)

The programmer's name parameter identifies the person or group responsible for a job.

programmer's name

Rules for Coding

1. If you code the programmer's name parameter, place it after the accounting information parameter or after the comma indicating its absence.
2. Code the programmer's name parameter before any or all keyword parameters.
3. The programmer's name must not exceed 20 characters, including all special characters.
4. When the programmer's name contains special characters, other than periods, enclose the name in apostrophes. If the special characters include apostrophes, each apostrophe must be coded as two consecutive apostrophes.
5. If the programmer's name parameter is not required, you need not code a comma to indicate its absence.

Examples of the Programmer's Name Parameter

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

Programmer's name; no accounting information.

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

Programmer's name containing special characters; no accounting information.

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

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

ADDRSPC Parameter

Keyword, Optional

The ADDRSPC parameter can be used to ensure that a program's virtual addresses are exactly the same as its real addresses. ADDRSPC is a mnemonic for *address space*.


For further information on the ADDRSPC parameter see *OS/VSI JCL Services*, GC24-5100.

$$\text{ADDRSPC} = \left\{ \begin{array}{l} \text{VIRT} \\ \text{REAL} \end{array} \right\}$$

Rules for Coding

1. Code the REGION parameter to specify how much real storage you need.
2. **Default:** If you omit the ADDRSPC parameter, VIRT is assumed unless the installation has changed the default in the reader procedure.

If you specify ADDRSPC=REAL and omit the REGION parameter, the default region size for the installation is assumed.

3.  If you specify the ADDRSPC parameter on the JOB statement, any ADDRSPC parameters on subsequent EXEC statements are ignored and the value on the JOB statement is used.

Examples of the ADDRSPC Parameter

```
//PEH      JOB      ADDRSPC=REAL,REGION=100K
```

The ADDRSPC parameter requests real storage. The REGION parameter specifies the amount, in this case, 100K.

```
//DEB      JOB      ADDRSPC=VIRT
```

The ADDRSPC parameter requests virtual storage.

CLASS Parameter

Keyword, Required

The CLASS parameter assigns a job class to each job, depending on the characteristics of the job and the installation's rules for assigning a job class.

For further information on the use of the CLASS parameter, see *OS/VS1 JCL Services, GC24-5100*.

CLASS=jobclass

Rules for Coding

1. Code any alphabetic character (A-Z) or numeric character (0-9), depending on the installation's rules for assigning a jobclass.
2. The CLASS parameter may be overridden by the PROFILE parameter. For further information, see the section on the PROFILE parameter.
3. **Default:** If you omit the CLASS parameter and the PROFILE parameter is not used, jobclass A or installation assigned default is assumed.

Examples of the CLASS Parameter

```
//SETUP      JOB      CLASS=C
```

Assigns a job to job class C.

```
//COSCO      JOB      CLASS=M, PRTY=10
```

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

COND Parameter

Keyword, Optional

The COND parameter specifies whether a job is to continue processing based on return codes issued by one or more of its job steps. Each test specified by the COND parameter is performed using the return code of a completed job step. If any of the tests are satisfied, the remaining jobsteps are bypassed or the job is terminated.

For further information on the use of the COND parameter, see *OS/VS1 JCL Services*, GC24-5100.

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. Operators and their meaning 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. You can code up to eight different return code tests for each job.
2. If you code only one return code test, you need not code the outer parentheses.
3. The COND parameter can also be coded on an EXEC statement. When a return code test requested on an EXEC statement is satisfied, the associated job step is bypassed.
4. 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 take 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.

Examples of the COND Parameter

```
//TYPE      JOB      COND=( 17,LT)
```

If 17 is less than the return code, the job is terminated. (Any return code less than or equal to 17 allows the job to continue.)

```
//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.)

MPROFILE Parameter

Keyword, Optional

The MPROFILE parameter is one of the selection parameters specified by the installation that, when coded in accordance with the format used by your installation, will assign a message class.

For further information, see *OS/VS1 Planning and Use Guide*, GC24-5090.

MPROFILE='message profile string'

'message profile string'

the information contained in the message profile string should describe your job's requirements. The format of the string will be established for your installation by the system programmer.

Rules for Coding

1. The MPROFILE parameter overrides the MSGCLASS parameter.
2. The message profile string must be enclosed in apostrophes. If the profile string continues from one line to another, each line must begin and end with an apostrophe and the entire group must be enclosed in parentheses.
3. The total length of the message profile string must not exceed 120 characters.

Examples of the MPROFILE Parameter

```
//HIGH      JOB      MPROFILE='FORM=SINGLE,PAGES=15'
```

The user has indicated his job's requirements. The system assigns a message class based on this information when compared to the table created by the system programmer for the installation.

```
//CAQ      JOB      MPROFILE=( 'FORM=MULTI' ,  
//          'PAGES=10,RUN=TEST' )
```

This example shows the method of continuing the MPROFILE parameter.

MSGCLASS Parameter

Keyword, Optional

The MSGCLASS parameter specifies the output class to which system messages are to be routed.

For further information on the use of the MSGCLASS parameter, see *OS/VSI JCL Services*, GC24-5100.

MSGCLASS=output class

Rules for Coding

1. The output class is an alphabetic (A-Z) or numeric (0-9) character.
2. **Default:** If you do not code the MSGCLASS parameter, the system messages associated with your job will be routed to the default output class specified by the installation. If the installation has not specified a default value, the default for the MSGCLASS parameter is A.
3. System messages and output data sets can be routed to the same output class. Code 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.
4. The MSGCLASS parameter may be overridden by the MPROFILE parameter. For further information, see the section on the MPROFILE parameter.

Examples of the MSGCLASS Parameter

```
//IN          JOB      MSGCLASS=F
```

Specifies an output class.

```
//BOTLE      JOB
```

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

```
//A1403      JOB      MSGCLASS=L  
//STEP1     EXEC     PGM=PRINT  
//OUTPUT    DD       SYSOUT=L
```

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

MSGLEVEL Parameter

Keyword, Optional

The MSGLEVEL parameter indicates what job output is to be written as part of the output listing.

For further information on the MSGLEVEL parameter, see *OS/VSI JCL Services*, GC24-5100.

MSGLEVEL=(statements,messages)

statements

a number that indicates which job control statements are to be written as output from a job.

The choices are:

- 0 only the JOB statement is to be written.
- 1 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 only input job control statements are to be written.

messages

a number that indicates which allocation/termination messages are to be written.

The choices are:

- 0 no allocation/termination messages are to be written, unless the job terminates abnormally.
- 1 all allocation/termination messages are to be written.

Rules for Coding

1. **Default:** If you do not code MSGLEVEL, default values established by the installation for both of the subparameters (statements, messages) are assumed by the system. Code the MSGLEVEL parameter only when the default value for either or both of the subparameters will not provide the desired output.
2. If you omit the statement subparameter, code a comma to indicate its absence.
3. If you omit the messages subparameter, you need not code the parentheses.

Examples of the MSGLEVEL Parameter

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

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

```
//STEL      JOB      MSGLEVEL=( 0, 1 )
```

Requests that only the JOB statement and all allocation/termination messages be written.

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

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

```
//PAUL      JOB      MSGLEVEL=0
```

Requests that only the JOB statement be written.

PROFILE Parameter

Keyword, Optional

The PROFILE parameter is one of the selection parameters specified by the installation that, when coded in accordance with the format used by your installation, will assign input class and priority.

For further information, see *OS/VSI Planning and Use Guide*, GC24-5090.

PROFILE='profile string'

'profile string'

the information contained in the profile string should describe the requirements of the job. The format of the string will be established by the system programmer for your installation.

Rules for Coding

1. The PROFILE parameter overrides the CLASS and PRTY parameters.
2. The profile string must be enclosed in apostrophes. If the profile string continues from one line to another, each line must begin and end with an apostrophe and the entire group must be enclosed in parentheses.
3. The total length of the profile string must not exceed 120 characters.

Example of the PROFILE Parameter

```
//RING      JOB      PROFILE='RUN=TEST,LANG=COBOL,TIME=10'
```

The user has indicated his job's requirements. The system assigns an input class and/or initiation priority based on this information when compared to the table created by the installation's system programmer.

```
//SMQ      JOB      PROFILE=( 'RUN=TEST' ,  
//          'LANG=FORT,TIME=5' )
```

This example shows the method of continuing the PROFILE parameter.

PRTY Parameter

Keyword, Optional

The PRTY parameter specifies a job's initiation priority within its job class. (The job class is assigned by the CLASS or PROFILE parameter on the JOB statement.)

For further information on the use of the PRTY parameter, see *OS/VS1 JCL Services*, GC24-5100.

PRTY=priority

priority

a number that specifies a priority of 0 to 13. The highest priority allowed is 13.

Rules for Coding

1. **Default:** If you do not code the PRTY parameter, the default priority specified by the installation in the cataloged procedure for the input reader is used as the priority value for the job.
2. Avoid using 13 as a priority value because it is used by the system to expedite processing of jobs in which certain errors have been diagnosed.
3. The PRTY parameter may be overridden by the PROFILE parameter. For further information, see the section on the *PROFILE* parameter.

Examples of the PRTY Parameter

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

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

```
//RING      JOB      PRTY=4
```

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

RD Parameter

Keyword, Optional

The RD (restart definition) parameter specifies that the step restart facilities are to be used to suppress the action of the CHKPT macro instruction and to suppress automatic step restart.

For detailed information on the checkpoint/restart facility, see *OS/VS1 Checkpoint/Restart*, GC26-3876.

$$\text{RD} = \left(\begin{array}{c} \text{R} \\ \text{RNC} \\ \text{NC} \\ \text{NR} \end{array} \right)$$

R

indicates that automatic step restart is permitted.

If the processing program used by a job step *does not* include any CHKPT macro instruction, coding RD=R allows execution to be resumed at the beginning of the abnormally terminated step.

If the program *does* include a CHKPT macro instruction, coding RD=R permits automatic step restart to occur 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.

RNC

indicates that automatic step restart is permitted and automatic checkpoint restart is not permitted. Deferred checkpoint restart also is not permitted.

NC

indicates that neither automatic step restart nor automatic checkpoint restart is permitted. Deferred checkpoint restart also is not permitted.

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, (on the job statement of the resubmitted job) the checkpoint at which execution is to be resumed.

Rules for Coding

1. If automatic step restart is permitted, (that is, when you have coded RD=R or RD=RNC), assign each job step a unique step name.

2. Code the RD parameter on the EXEC statement instead of the JOB statement when you want to make different restart requests for each job step.
3. If you code the RD parameter on the JOB statement, any RD parameters coded on the job's EXEC statements are ignored and the value coded on the JOB statement is effective for all steps.
4. The RD parameter is ignored for system tasks and generalized start jobs.

Examples of the RD Parameter

```
//JILL      JOB      RD=R
```

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

```
//TRY56     JOB      RD=RNC
```

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

```
//PASS      JOB      RD=NR
```

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

REGION Parameter

Keyword, Optional

The REGION parameter specifies the amount of real storage to be allocated to a job.

For further information on the REGION parameter, see *OS/VS1 JCL Services*, GC24-5100.

REGION=valueK

valueK

a number that indicates how many bytes of storage are to be allocated to a job.

Rules for Coding

1. Code an even number. If you code an odd number, the system treats it as the next highest even number. Do not code REGION=0K; it causes a JCL error.
2. **Default:** If you omit the REGION parameter, the REGION size default value in the input reader procedure is used.
3. When you want to specify a different region size for each job step, code the REGION parameter on the EXEC statements, instead of on the JOB statement.
4. If you code the REGION parameter on the JOB statement, REGION parameters coded on the job's EXEC statements are ignored.
5. If you code ADDRSPC=VIRT with the REGION parameter, REGION is ignored.

Example of the REGION Parameter

```
//CAC      JOB      143,ADDRSPC=REAL,REGION=40K
```

Requests 40K of real storage space to be allocated to this job.

RESTART Parameter

Keyword, Optional

The RESTART parameter indicates that the restart facilities are to be used to resubmit a job for execution. Execution can be restarted at the beginning of a step (step restart) or within a step (checkpoint restart).

For detailed information on the checkpoint/restart facilities, refer to *OS/VS1 Checkpoint/Restart*, GC26-3876.

$$\text{RESTART}=(\left. \begin{array}{l} * \\ \text{stepname} \\ \text{stepname.procstepname} \end{array} \right\} [\text{checkid}])$$

*

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.

checkid

specifies 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. Code * in place of stepname.procstepname if the first job step calls a cataloged procedure and execution is to be restarted at or within the first procedure step.
2. You need not code the parentheses if execution is to be restarted at a job step, that is, if you do not code the checkid subparameter.
3. 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.
4. Include the SYSCHK DD statement when execution is to be restarted within a job step. (The SYSCHK DD statement is described in the section on DD statements in this publication.)

5. 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,...).
6. Generation data sets: In the restart step or in steps following it, do not use your original relative generation numbers to refer to generation data sets that were created and cataloged in steps preceding the restart step. Instead, refer to a generation data set by its present relative generation number. For example, if the last generation data set created and cataloged was assigned a generation number of +2, refer to it as 0 in the restart step and in steps following the restart step. In this case, refer to the generation data set assigned a generation number of +1 as -1.

If generation data sets created in the restart step were kept instead of cataloged (for example, DISP=(NEW,CATLG,KEEP) was coded and abnormal termination occurred), refer to these data sets and generation data sets during checkpoint restart, by the same relative generation numbers that were used to create them.

7. For a 3800 at checkpoint/restart, the job has the JFCBE parameter as modified during the last JFCBE exit. If no exit was requested or if the JFCBE was not flagged as being modified during the exit, the JFCBE reflects the values coded in the restart JCL. For additional information on the JFCBE exit, see *OS/VSI Data Management Services Guide*.

Examples of the RESTART Parameter

```
//LINES      JOB      RESTART=COUNT
```

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

```
//@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 on which an entry for the checkpoint name CHKPT3 was written.

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

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

```
//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.

TIME Parameter

Keyword, Optional

The TIME parameter specifies the maximum amount of time that a job may use the CPU. The CPU time used by the job is written on the output listing.

For further information on the use of the TIME parameter and its relation to the SMF (System Management Facilities) feature, see *OS/VS1 System Management Facilities (SMF)*, GC24-5115.

$$\text{TIME} = \left\{ \begin{array}{l} \text{([minutes],[seconds])} \\ 1440 \end{array} \right\}$$

minutes

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

seconds

a number that 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.

Rules for Coding

1. If you code the CPU time limit in minutes only, you need not code the parentheses.
2. If you code the CPU time limit in seconds only, you must code a comma preceding the seconds to indicate the absence of minutes.
3. Code 1440 if the job may use 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.
4. Code the TIME parameter on EXEC statements if you want to indicate how long each step can use the CPU.
5. When you do not code the TIME parameter on the JOB statement or if you code TIME=0, no CPU time limit is assigned to a job. However, each job step is still timed due to the step time default specified in the parm field in the reader procedure.
6. *System Management Facilities*: Normally, a job that exceeds the time limit specified in the TIME parameter is terminated. However, if SMF is in use, a user exit routine can extend the time limit so that processing continues.

Without SMF in use, 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 terminates the job.

With SMF in use in the system, the installation determines the time limit for wait states through the use of the JWT parameter of SMF. A job step remaining in a wait state for more than the established time limit terminates the job unless a user-provided exit routine extends the wait state limit for that step.

Without SMF in use, the time limit is determined at system generation time by the WAIT parameter of the CTRLPROG macro. A value of 30 is the default if WAIT is not specified. A job step remaining in a wait state for more than 30 consecutive minutes is terminated.

Examples of the TIME Parameter

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

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

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

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

```
//FORMS     JOB      TIME=5
```

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

```
//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.

TYPRUN Parameter

Keyword, Optional

The TYPRUN parameter specifies that a job is to be held for execution until some event has occurred. The operator must be informed of what event is to occur. When the event has occurred, the operator may issue a RELEASE command, thereby allowing the job to be selected for processing.

The TYPRUN parameter can also specify that the JCL for a job be scanned for syntax errors.

For further information on the TYPRUN parameter, see *OS/VSI JCL Services*, GC24-5100.

$$\text{TYPRUN} = \left\{ \begin{array}{l} \text{HOLD} \\ \text{SCAN} \end{array} \right\}$$

HOLD

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

SCAN

specifies that the JCL for a job is to be scanned for syntax errors, but that the job is not to be executed.

Example of the TYPRUN Parameter

Jobs UPDATE and LIST are to be submitted for execution. The job UPDATE uses a program that adds members to and deletes members from a library; the job LIST uses a program that lists the members of a library. For 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 MONITOR 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

Control Statement

The EXEC statement is the first statement of each job step and cataloged procedure step. It identifies the program to be executed or the cataloged procedure to be called.

```
┌ //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

1. Code an EXEC statement for each job step.
2. A job cannot contain more than 255 job steps and procedure steps. If the job requests a deferred checkpoint restart, it cannot contain more than 254 job steps and procedure steps.
3. A stepname is optional. However, when you want to perform certain functions, you must code a valid and unique stepname in the name field for each job step within the job. A stepname is necessary to:

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

The stepname must consist of one through eight alphameric and national (@, #, \$) characters. The first character must be an alphabetic or national character.

4. The two types of parameters that can be coded in the operand field of the EXEC statement are positional and keyword.

Positional parameters must precede any keyword parameter. One of the following two positional parameters can be coded:

```
PGM  
PROC
```

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

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

```
ACCT  
ADDRSPC  
COND  
PARM  
RD  
REGION  
TIME
```


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

Sample EXEC Statements

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

```
// EXEC PGM=ENTRY, TIME=(2,30)
```

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

PGM Parameter

Positional, Optional

The PGM parameter specifies a program to be executed. The specified program must be a member of a temporary library, the system library, or a private library.

For further information on identifying programs and on libraries (partitioned data sets), see Appendix A of this publication.

$$\text{PGM} = \left\{ \begin{array}{l} \text{program name} \\ \text{*stepname.ddname} \\ \text{*stepname.procstepname.ddname} \end{array} \right\}$$

program name

the member name or alias of the program to be executed.

*.stepname.ddname

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; ddname is the name that appears on the DD statement.

This form of the parameter is usually used when a previous job step has created a temporary partitioned data set to store a program until the program is required.

*.stepname.procstepname.ddname

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; procstepname is the name of the procedure step that contains the DD statement.

Rules for Coding

1. If you code the PGM parameter, code it as the first parameter on the EXEC statement. The program you specify must be a member of a partitioned data set.
2. The program name must have one to eight alphanumeric or national characters, of which the first must be alphabetic or national.

Examples of the PGM Parameter

```
//STEP1 EXEC PGM=TABULATE
```

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

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

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

```
//CREATE    EXEC     PGM=IEWL
//SYSLMOD   DD      DSNNAME=εεPARTDS( PROG ),UNIT=2314,
//          DISP=( MOD,PASS ),
//          SPACE=( 1024,( 50,20,1 ) )
//EXECUTE   EXEC     PGM=*.CREATE.SYSLMOD
```

Uses a backward reference to a DD statement that defines a temporary library created in the step named CREATE. The program named PROG is stored as a member of the partitioned data set named εεPARTDS and is executed in the step named EXECUTE. εεPARTDS is deleted at the end of the step named EXECUTE.

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

Uses a 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.

```
//CHECK     EXEC     PGM=IEFBR14
```

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

Identifying the Program to be Executed

All executable programs 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 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 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 subsequent job step. When the program is required, you can 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 can 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.

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 that contains frequently used programs, as well as system programs. You can request the use of a program that is a member of the system library by coding PGM=program name. The system automatically looks in SYS1.LINKLIB for a member with that name.

A program that resides in the system library can 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 can request the 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.

A program that resides in a private library can 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.

IEFBR14 Program

If space allocation and disposition processing requests are contained in your job control statements, you can satisfy these requests before executing your program. To do this, substitute IEFBR14 for the program name. This also allows you to check the accuracy of your statements. (If you create a data set when using this program, the data set's status will be OLD when you execute your own program.)

PROC Parameter

Positional, Optional

The PROC parameter specifies that a cataloged procedure or an in-stream procedure is to be called and executed.

For further information on cataloged and in-stream procedures, see *OS/VS1 JCL Services*, GC24-5100.

$$\left\{ \begin{array}{l} \text{PROC=procedure name} \\ \text{procedure name} \end{array} \right\}$$

procedure name

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

Rules for Coding

1. The procedure name must consist of one through eight alphameric or national (@, #, \$) characters, of which the first must be alphabetic or national.
2. If you code the PROC parameter, code it as the first parameter on the EXEC statement, instead of the PGM parameter. You may code only the cataloged or in-stream procedure name, omitting PROC.
3. When the EXEC statement specifies a cataloged or in-stream procedure, parameters in the operand field of the EXEC statement will override EXEC parameters in the called procedure.
4. Any DD statements that follow the EXEC statement are treated as overriding DD statements or DD statements that are to be added to the cataloged or in-stream procedure for the duration of the job step.

Examples of the PROC Parameter

```
//SP3      EXEC      PROC=PAYWKRS
```

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

```
//BK       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.

ACCT Parameter

Keyword, Optional

The ACCT parameter specifies one or more subparameters of accounting information to be passed to the installation's accounting routines.

For further information concerning the accounting routines, see *OS/VSI Planning and Use Guide, GC24-5090*.

ACCT=(accounting information[, . . .])

accounting information

specifies one or more subparameters established by the installation as accounting information to be passed to the accounting routines.

Rules for Coding

1. If the accounting information consists of only one subparameter, you can omit the parentheses.
2. If the accounting information consists of several subparameters, separate each subparameter by a comma.
3. The maximum number of characters of accounting information, including the commas that separate the subparameters, is 142.
4. If a subparameter contains special characters (other than hyphens), 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.
5. If the job step calls a cataloged procedure, the ACCT parameter overrides any ACCT parameters coded in the procedure steps. This pertains to all procedure steps.
6. If different steps in a cataloged procedure require different accounting information, code `ACCT.procstepname=(accounting information,...)` for each step that requires unique accounting information. Accounting information then pertains only to the named procedure step.

Examples of the ACCT Parameter

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

Specifies that this accounting information pertains to this job step.

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

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

//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).

ADDRSPC Parameter

Keyword, Optional

The ADDRSPC parameter can be used to ensure that a program's virtual addresses are exactly the same as its real addresses. ADDRSPC is a mnemonic for *address space*.

For further information on the ADDRSPC parameter, see *OS/VSI JCL Services*, GC24-5100.

$$\text{ADDRSPC} = \left\{ \begin{array}{l} \text{VIRT} \\ \text{REAL} \end{array} \right\}$$

Rules for Coding

1. Code the REGION parameter to specify how much real storage you need.
2. **Defaults:** If you omit the ADDRSPC parameter, VIRT is assumed unless the installation has changed the default in the reader procedure.

If you specify ADDRSPC=REAL and omit the REGION parameter, the default region size for the installation is assumed.

3. If you specify the ADDRSPC parameter on the JOB statement, any ADDRSPC keyword parameter on subsequent EXEC statements are ignored and the value on the JOB statement is used.

Examples of the ADDRSPC Parameter

```
//CAC1      EXEC    A,ADDRSPC=REAL,REGION=80K
```

The ADDRSPC parameter requests real storage. The REGION parameter specifies the amount; in this case, 80K.

```
//CAC2      EXEC    B,ADDRSPC=VIRT
```

The ADDRSPC parameter requests virtual storage.

COND Parameter

Keyword, Optional

The COND parameter specifies whether or not a job step will be executed based on return codes issued by one or more of the preceding job steps. This parameter allows the specification of conditions for bypassing a job step, as well as for executing a job step.

You can use the COND parameter to test the return codes which are issued by the compiler, assembler, and linkage editor programs. 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.

For further information on use of the COND parameter, see *OS/VS1 JCL Services*, GC24-5100.

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

code

a 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. 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 preceding job steps have abnormally terminated. If return code tests are to be made and any tests are satisfied, this job step is bypassed.

ONLY

specifies that the job step is to be executed only if one or more preceding job steps have abnormally terminated. If the current job step specifies that return code tests are to be made and any tests are satisfied, this job step is bypassed.

Note: 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 you specify neither, the job step is bypassed and the EXEC statement of the next step is scanned for the EVEN or ONLY subparameter. If you specify EVEN or ONLY, return code tests, if any, are made on all previous steps specified that executed and did not abnormally terminate. If any one of these tests is satisfied, the step is bypassed. Otherwise, the job step is executed.

Rules for Coding

1. If you do not code EVEN or ONLY, you can make up to eight tests on return codes issued by preceding job steps or cataloged procedure steps, which completed normally. If you code EVEN or ONLY, you can make up to seven tests on return codes.
2. If you code only EVEN or ONLY, or if you code only one test, you can omit the outer parentheses.
3. If you want each return code test to be made on the return code issued by every preceding step, do not code a stepname.
4. You can code the EVEN or ONLY subparameters before, between, or after return code tests.
5. When you code the COND parameter 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.
6. *Backward references:* 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) does not exist if the step creating it was bypassed, or (2) may be incomplete if the step creating it abnormally terminated. If the job step refers the system to an earlier job step for volume and unit information, this information may not be available if the earlier job step was bypassed.
7. *Overriding COND parameters:* The COND parameter may be coded on an EXEC statement that calls a cataloged procedure. If a 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 subparameters 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.
 - To override the COND parameter on an EXEC statement that calls a cataloged or instream procedure and,

- 1) the condition code being tested is from a previous step in the procedure or,
- 2) the condition code being tested is from another procedure already executed, code:
COND.procstepname=(code,operator,stepname.procstepname).

Examples of the COND Parameter

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

If 4 is greater than the return code issued by STEP3, the system bypasses this step. (A return code of 4 or greater allows this step to be executed.) Because neither EVEN nor ONLY is specified, this job step is automatically bypassed if a preceding step abnormally terminates.

```
//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 (any return code of 17 through 89 does not satisfy the tests) and a preceding job step has abnormally terminated, this step is executed because ONLY is coded.

```
//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. Because neither EVEN nor ONLY is specified, this job step is automatically bypassed if a preceding step abnormally terminated.

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

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

PARM Parameter

Keyword, Optional

The PARM parameter passes variable information to a program at execution time.

For further information on the PARM parameter, see *OS/VSI Supervisor Services and Macro Instructions*, GC24-5103.

PARM=value

value

up to 100 characters of information that the system passes to a processing program.

Rules for Coding

1. If the value contains more than one expression separated by commas, you must enclose it in apostrophes or parentheses, for example, PARM='P1,123,MT5' or PARM=(P1,123,MT5).

The 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 you include special characters in any expression, either (1) enclose the value in apostrophes, or (2) enclose the expression in apostrophes and the value in parentheses, for example, 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 ampersand and you are not defining a symbolic parameter, you should code two consecutive ampersands in its place, for example, PARM='3462&&5'.

When two apostrophes or two ampersands are coded, only one is passed to the processing program.

3. If you must continue the value on another statement, enclose it in parentheses. The continuation comma is considered part of the value field and counts toward the maximum of 100 characters of data. You may not continue any value enclosed in apostrophes.
4. If a 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 nullify the PARM parameters in the procedure, code the PARM parameter on the EXEC statement that calls the procedure. The information contained in the PARM parameter passes 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 in the PARM value is passed only to the named procedure step.

Examples of the PARM Parameter

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

Passes the information in the PARM parameter, except the apostrophes and the parentheses, to the processing program named APG22.

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

Passes this information to the first step of the procedure named PROC81. If any of the other procedure steps in PROC81 contain the PARM parameter, those parameters are nullified.

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

Passes this information to the procedure step named LKED. If any of the other steps in the procedure ASMFCLG contain the PARM parameter, those parameters are still effective.

RD Parameter

Keyword, Optional

The RD (restart definition) parameter specifies that the step restart facilities are to be used to suppress the action of the CHKPT macro instruction and to suppress automatic restarts.

For detailed information on the checkpoint/restart facilities, see *OS/VS1 Checkpoint/Restart*, GC26-3876.

$$\text{RD} = \left(\begin{array}{l} \text{R} \\ \text{RNC} \\ \text{NC} \\ \text{NR} \end{array} \right)$$

R

indicates that automatic step restart is permitted.

If the processing program used by a job step does not include any CHKPT macro instructions, coding RD=R allows execution to be resumed at the beginning of the abnormally terminated step.

If the program includes a CHKPT macro instruction, coding RD=R permits automatic step restart to occur 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.

RNC

indicates that automatic step restart is permitted and automatic checkpoint restart is not permitted. Deferred checkpoint restart also is not permitted.

NC

indicates that neither automatic step restart nor automatic checkpoint restart is permitted. Deferred checkpoint restart also is not permitted.

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, (on the job statement of the resubmitted job) the checkpoint at which execution is to be resumed.

Rules for Coding

1. If automatic step restart is permitted, for example, RD=R or RN=RNC is coded, assign each job step a unique step name.

2. Code the RD parameter on the EXEC statements instead of the JOB statement when you want to make different restart requests for each job step.

If you code the RD parameter on the JOB statement, any RD parameters coded on the job's EXEC statements are ignored and the value coded on the JOB statement is effective for all steps.

3. The RD parameter can be coded on the EXEC statement of a cataloged procedure. If the job step calls a cataloged procedure:

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 then pertains only to the named procedure step.

4. The RD parameter is ignored for system tasks and generalized start jobs.

Examples of the RD Parameter

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

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

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

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

```
//CARD      EXEC      PGM=WTE, 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.

```
//STP4      EXEC      BILLING, RD.PAID=NC, RD.BILL=NR
```

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

REGION Parameter

Keyword, Optional

The REGION parameter specifies the amount of real storage to be allocated to a job step.

For further information on the REGION parameter, see *OS/VS1 JCL Services*, GC24-5100.

REGION=valueK

valueK

a number that indicates how many bytes of storage are to be allocated to a job step.

Rules for Coding

1. Code an even number. (If you code an odd number, the system treats it as the next highest even number.) Do not code REGION=0K; it will cause a JCL error.
2. **Default:** If you omit the REGION parameter, the REGION size default value in the input reader procedure is used.
3. If you code the REGION parameter on the JOB statement, REGION parameters on the job's EXEC statements are ignored.
4. If you code ADDRSPC=VIRT with the REGION parameter, REGION is ignored.

Example of the REGION Parameter

```
//MKBOYLE EXEC A,ADDRSPC=REAL,REGION=40K
```

Requests 40K of real storage space to be allocated to this job step.

TIME Parameter

Keyword, Optional

The TIME parameter specifies the maximum amount of time that a job step may use the CPU. The CPU time used is written on the output listing.

For further information on the use of the TIME parameter, see *OS/VS1 System Management Facilities (SMF)*, GC24-5115.

$$\text{TIME} = \left\{ \begin{array}{l} (\text{[minutes],[seconds]}) \\ 1440 \end{array} \right\}$$

minutes

a number that 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

a number that 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, unless there is a job time limit specified on the JOB card. The remaining job time is used for timing the step.

Rules for Coding

1. If you code the CPU time limit in minutes only, you can omit the parentheses.
2. If you code the CPU time limit in seconds only, you must code a comma preceding the seconds to indicate the absence of minutes.
3. Code 1440 if the job step may use the CPU for 24 hours or more, or if job step should be allowed to remain in a wait state for more than the established time limit.
4. TIME=0 causes the job step to ABEND.
5. Code the TIME parameter on the JOB statement if you want to indicate how long the entire job can use the CPU.
6. You may code the TIME parameter on the EXEC statement of a cataloged procedure step.

To override *all* TIME parameters in a cataloged procedure, code the TIME parameter on the EXEC statement that calls the procedure. This establishes 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.

7. **Default:** When you do not code the TIME parameter on the EXEC statement, a default time limit, specified by the installation in the reader procedure, is assumed for each job step.
8. 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 or less than the step default time assigned in the reader procedure, the step can use the CPU only for the job's remaining CPU time.
9. Normally, a step that exceeds the specified time limit terminates the job. However, if SMF (System Management Facilities) is in use in the system, a user exit routine can extend the time limit so that processing continues.

Without SMF in use, 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 terminates the job.

With SMF in use, the installation determines the time limit for wait states through the use of the JWT parameter of SMF. A job step remaining in a wait state for more than the established time limit terminates the job unless a user-provided exit routine extends the wait state limit for that step.

Without SMF in use, the time limit is determined at system generation time by the WAIT parameter of the CTRLPROG macro. A value of 30 is the default if WAIT is not specified. A job step remaining in a wait state for more than 30 consecutive minutes is terminated.

Examples of the TIME Parameter

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

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

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

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

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

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

```
//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.

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

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

Section IV: The DD Statement

Control Statement

The DD (data definition) statement describes a data set to be used in a job step and specifies the input and output facilities required for the data set.

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

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

Rules for Coding

1. Code a DD statement for each data set to be used in a step.
2. Code a ddname, beginning in column 3, and consisting of one through eight alphanumeric and national (@, #, \$) characters. The first character must be alphabetic or national.
3. Code unique ddnames within a job step. If duplicate ddnames exist in a step, 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.
4. Apart from the restricted use of certain special ddnames (listed below), you should not code a ddname at all in the two instances:
 - a. If a DD statement is to define a data set that is concatenated with a data set defined by a preceding DD statement.
 - b. If the DD statement is the second or third consecutive DD statement that defines an indexed sequential data set.
5. Special ddnames: Do not use the following five special ddnames unless you want to make use of the particular facilities that these names represent to the system. These facilities are explained in detail in the following pages.

```
JOBCAT ( for VSAM only )  
JOB LIB  
STEP CAT ( for VSAM only )  
STEP LIB  
SYS ABEND  
SYS DUMP  
SYS CHK
```

6. 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.
7. The maximum number of DD statements allowed per job step is 255.

8. Two types of parameters can be coded on a DD statement: *keyword* and *positional*. The *positional* parameters, which must precede any keyword parameters, are:

```
*
DATA
DUMMY
```

The *keyword* parameters are:

AFF	DDNAME	HOLD	SPLIT
AMP(VSAM only)	DEST	LABEL	SUBALLOC
BURST	DISP	MODIFY	SUBSYS
CHARS	DLM	MSVGP	SYSOUT
CHKPT	DSID	OUTLIM	TERM
COMPACT	DSNAME	QNAME	UCS
COPIES	FCB	SEP	UNIT
DCB	FLASH	SPACE	VOLUME

Rules for Coding DD Statements when Using Cataloged Procedures

1. If a job step uses a cataloged procedure, you may make modifications to the DD information within the procedure for the duration of the job step. To do this, code modifications on the DD statements immediately following the EXEC statement used to call the cataloged procedure.
2. To override parameters on a DD statement within a cataloged procedure, code the name of the procedure step in which the DD statement appears, followed by a period, and followed by the name of the DD statement that you want to override. When overriding two or more DD statements in a procedure step, the sequence of the overriding statements must be the same as the sequence of the procedure statements being overridden.
3. To add DD statements to a procedure step, code the name of the procedure step in which the DD statement appears, followed by a period, and followed by a ddname of your choice. Added statements must follow all overriding statements for the procedure step.
4. 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 must be the name coded in the DDNAME parameter.

Examples of Valid DDnames

```
//INPUT DD
// DD
```

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

```
//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.

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

This sequence defines data sets that are to be concatenated and added to the procedure step. On the first DD statement, the procedure step to which statements are to be added is identified and followed by any valid ddname. On the second DD statement, the ddname is omitted.

JOBCAT Facility (VSAM only)

DD Statement

The JOBCAT (job catalog) DD statement appends a VSAM user catalog to the VSAM master catalog for the duration of job execution. This facility provides access to VSAM component, cluster, and volume information that resides in the VSAM user catalog referenced in the JOBCAT DD statement. It also provides access to non-VSAM data sets cataloged in the VSAM user catalog.

For additional information about the JOBCAT facility, see *OS/VS Virtual Storage Access Method (VSAM) Programmer's Guide, GC26-3838*.

Rules for Coding

1. The ddname on the statement must be JOBCAT.
2. The JOBCAT DD statement must appear after the JOB statement, but before the first EXEC statement.
3. If a JOBLIB statement is used, it must precede the JOBCAT statement.
4. A job catalog is similar to the JOBLIB data set except that a job catalog contains information about VSAM components, VSAM clusters, and volumes containing VSAM data sets.
5. Several user catalogs can be concatenated for the JOBCAT statement.

Example of the JOBCAT DD Statement

```
//PAYROLL JOB  
//JOBLIB DD DSNAME=PRIVATE.LIB4,  
// DISP=(OLD,PASS)  
//JOBCAT DD DSN=USER.JOBCAT,DISP=OLD  
//STEP1 EXEC PGM=SCAN  
//STEP2 EXEC PGM=UPDATE
```

This is an example of the placement of the JOBCAT DD statement.

JOBLIB Facility

DD statement

The JOBLIB DD statement defines a private library made available by the system to an entire job.

Rules for Coding

General

1. Code JOBLIB as the ddname on the first DD statement. Never use the ddname JOBLIB except to define a private library for an entire job.

Omit the ddname from all subsequent DD statements that define data sets that are to be concatenated to the first one. These DD statements must immediately follow the JOBLIB statement, and the JOBLIB statement must immediately follow the JOB statement.

2. If you include a JOBLIB DD statement in the JCL for a job, each time the job requests a program, the system searches the private library. If it does not find the program there, it searches the system library.

Use a STEPLIB DD statement, described under the *STEPLIB Facility*, to define a private library to be made available to one job step in a job. If you include a STEPLIB DD statement for a job step and a JOBLIB DD statement for the entire job, the system searches the step library and then the system library for the requested program. The job library is ignored for that step.

3. To make the private library available throughout the job, code the DISP parameter to specify the library's status and disposition.

For further explanation, refer to the DISP Parameter later in this section.

4. The rules for coding parameters on the JOBLIB DD statement depend on whether the private library is cataloged. These rules are discussed in the following text under separate headings.
5. Do not use a JOBLIB DD statement in a cataloged procedure.

When the Library Is Cataloged

1. Code the DSNNAME parameter to specify the name of the private library.
2. Code the DCB parameter if complete data control block information is not contained in the data set label.
3. *To refer to the private library in a later DD statement*, code DSNNAME=*.JOBLIB and the DISP parameter.

If a later DD statement defines a data set that is to be placed on the same volume as the private library, code `VOLUME=REF=*.JOBLIB` to obtain volume and unit information.

When the Library Is not Cataloged

1. Code the `UNIT` parameter to specify the device to be allocated to the library.
2. Code the `DSNAME` parameter unless the data set has been assigned a disposition of `(NEW,PASS)`.
3. Code the `VOLUME` parameter unless the status of the data set is `NEW`.
4. If the status of the data set is `NEW`, you must code the `SPACE` parameter to allocate space for the data set on the designated volume.
5. Code the `DCB` parameter if complete `DCB` information is not contained in the data set label.
6. *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,PASS)`.

If a later DD statement defines a data set that is to be placed on the same volume as the private library, code `VOLUME=REF=*.JOBLIB` to obtain volume and unit information.

Examples of the JOBLIB DD Statement

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

```
//PAYROLL JOB
//JOBLIB DD DSN=PRIV.DETP58,DISP=(OLD,PASS),
// UNIT=2314,VOLUME=SER=D58PVL
//STEP EXEC PGM=DAY
//STEP2 EXEC PGM=BENEFITS
//DD1 DD DSN=*.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.

```
//TYPE JOB MSGLEVEL=(1,1)
//JOBLIB DD DSN=GROUP8.LEVEL5,DISP=(NEW,
// CATLG),UNIT=2314,VOLUME=SER=148562,
// SPACE=(CYL,(50,3,4))
//STEP1 EXEC PGM=DISC
//DDA DD DSN=GROUP8.LEVEL5(RATE),DISP=OLD,
// VOL=REF=*.JOBLIB
//STEP2 EXEC PGM=RATE
```

The private library defined on the JOBLIB DD statement does not exist; 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, but (in STEP2) first looks for the program named RATE in the private library.

```
//PAYROLL JOB
//JOBLIB DD DSN=KRG.LIB12,DISP=(OLD,PASS)
// DD DSN=GROUP31.TEST,DISP=(OLD,PASS)
// DD DSN=PGMSLIB,UNIT=2314,
// DISP=(OLD,PASS),VOLUME=SER=34568
//STEP1 EXEC PGM=ARROW
//STEP2 EXEC PGM=BOW
```

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

STEPCAT Facility (VSAM only)

DD Statement

The STEPCAT (step catalog) DD statement appends a VSAM user catalog to the VSAM master catalog for the duration of a job step. This facility provides access to VSAM component, cluster, and volume information that resides in the VSAM user catalog referenced in the STEPCAT DD statement. It also provides access to nonVSAM data sets cataloged in the VSAM user catalog.

For additional information about the STEPCAT facility, see *OS/VS Virtual Storage Access Method (VSAM) Programmer's Guide*, GC26-3838.

Rules for Coding

1. The ddname on the statement must be STEPCAT.
2. The STEPCAT DD statement must appear after the EXEC statement, but it may appear anywhere in the set of DD statements for the step. The STEPCAT DD statement overrides the JOBCAT DD statement.
3. The step catalog is similar to the STEPLIB data set except that a step catalog contains information about VSAM components, VSAM clusters, and volumes containing VSAM data sets.
4. Several user catalogs can be concatenated for the STEPCAT statement.

Example of the STEPCAT DD Statement

```
//PAYROLL JOB  
//JOBLIB DD DSNAME=LIB5.GROUP3,  
// DISP=(OLD,PASS)  
//STEP1 EXEC PROC=SNZ12  
//STEPCAT DD DSNAME=USER.STEPCAT,  
// DISP=SHR
```

This example shows the placement of the STEPCAT DD statement.

STEPLIB Facility

DD Statement

The STEPLIB DD statement defines a private library to be made available by the system to a job step.

Rules for Coding

General

1. The ddname on this statement must be STEPLIB. Never use the ddname STEPLIB except to define a private library for a job step.
2. A STEPLIB DD statement can appear in any position among the DD statements for a step.
3. A private library defined on a STEPLIB DD statement can be referenced by, or passed on to, later job steps in the same job.
4. If you include a STEPLIB DD statement in the JCL for a job, when the jobstep for which the library is defined requests the program, the system first searches the private library; if it does not find the program there, it searches the system library.

Use a JOBLIB DD statement, described under the JOBLIB facility, to define a private library to be made available to an entire job. If you include a JOBLIB DD statement for the entire job *and* a STEPLIB DD statement for an individual job step, the system first searches the step library and then the system library for the requested program. The job library is ignored for that step.

5. A STEPLIB DD statement can appear in any cataloged procedure except those for readers, writers, initiators, DSO and RTAM.
6. To concatenate libraries, for example, to arrange a sequence of DD statements that define different data sets:
 - a. Code STEPLIB as the ddname on the first DD statement.
 - b. Omit the ddname from all subsequent DD statements that define private libraries for the particular step.
7. If you want the system to ignore the JOBLIB for a particular job step, use the following STEPLIB DD statement:

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

For the particular job step, the system first searches the system library for the required data set.

8. The rules for coding parameters on the STEPLIB DD statement depend on whether the library is cataloged, not cataloged, or passed by a previous job step. These rules are discussed in the following text under separate headings.

When the Library Is Cataloged

1. Code the DSNAME parameter to specify the name of the private library.
2. Code the DISP parameter to specify the library's status and its disposition. Its status must be either OLD or SHR. Its disposition may be any valid disposition.
3. Code the DCB parameter if complete DCB information is not contained in the data set label.

When the Library Has Been Passed by a Previous Step

1. Within a job, a defined step library can be made available for use by subsequent job steps by assigning a disposition of PASS.
2. To reference a previously defined step library:

Code the DSNAME parameter, specifying either the name of the step library or a backward reference of the form **.stepname.ddname*. If the step library was defined in a cataloged procedure, the backward reference must include the procedure step name; for example,
**.stepname.procstepname.ddname*

Code the DISP parameter, specifying a status of OLD and a disposition, depending on what you want done with the private library after its use in the job step.

3. Code the DCB parameter if complete DCB information is not contained in the data set label.

When the Library Is Neither Cataloged Nor Passed

1. Code the DSNAME parameter, specifying the name of the private library.
2. Code the DISP parameter, specifying the library's status, either OLD or SHR and a disposition, depending on what you want done with the private library after its use in the job step.
3. Code the VOLUME parameter, identifying the volume serial number.
4. Code the UNIT parameter, specifying the device to be allocated to the library.
5. Code the DCB parameter if complete DCB information is not contained in the data set label.

Examples of the STEPLIB DD Statement

```
//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.

```

//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),
// UNIT=2314,VOLUME=SER=55566
//STEP3 EXEC PGM=A1530
//STEP4 EXEC PGM=SNAP11
//STEPLIB DD DSNAME=* .STEP2: .STEPLIB,
// 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. Because a JOBLIB DD statement is included, STEP1 and STEP3 could execute programs from LIB5.GROUP4 or, if the programs are not found there, from SYS1.LINKLIB. STEP2 and STEP4 could execute programs from LIBRARYP or SYS1.LINKLIB.

```

//PAYROLL JOB
//JOBLIB DD DSNAME=LIB5.GROUP4,DISP=(OLD,PASS)
//STEP1 EXEC PGM=SUM
//STEPLIB DD DSNAME=SYS1.LINKLIB,DISP=SHR
//STEP2 EXEC PGM=VARY
//STEP3 EXEC PGM=CALC
//STEPLIB DD DSNAME=PRIV.WORK,DISP=(OLD,PASS)
// DD DSNAME=LIBRARYA,DISP=(OLD,KEEP),
// UNIT=2314,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 only a program from the system library, because the library defined on the STEPLIB DD statement is the system library. 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 it is not found there, in SYS1.LINKLIB.

SYSABEND and SYSUDUMP Facilities

DD Statements

The SYSABEND DD statement defines a data set on which a dump can be written if the step in which the statement appears abends. The dump provided by this facility includes the system nucleus, the processing program storage area, and a trace table.

The SYSUDUMP DD statement defines a data set on which a dump can be written if the step in which the statement appears abends. The dump provided by this facility includes only the processing program storage area.

You may request the SYSABEND or the SYSUDUMP data set to be printed in a high-density format on the 3800, directly or via SYSOUT. By printing 204 characters per line at 15 characters per inch, each print line contains twice as much data as that in a standard format dump. The high-density dump can also print at either 6 or 8 lines per inch.

For information on how to interpret dumps, see *OS/VSI Debugging Guide*, GC24-5093.

Rules for Coding

1. To dump to a unit record device, code:
 - The UNIT parameter, specifying the unit record device to which you want to write the dump, or
 - The SYSOUT parameter, specifying the output class through which you want the data set routed.
2. If you want to store the dump on a direct access device, code either the SPACE, SPLIT, or SUBALLOC parameter, specifying the amount of space you want allocated to the data set. If you want to specify a specific direct access device on which the dump is to be stored, also code the UNIT parameter. Otherwise, the system assigns a device and space for the dump.
3. If you want to store the dump and do not want it written immediately to any output device, code the following parameters:
 - The DSNNAME parameter, specifying the name of the data set.
 - The UNIT parameter, specifying the device to be allocated to the data set.
 - The VOLUME parameter, identifying the serial number of the volume to which the dump is to be written.
 - The DISP parameter, specifying status and disposition of the data set. Because you want to store the data set, make the conditional disposition KEEP or CATLG.
4. If you want the high-density dump, code CHARS=DUMP on the DD statement. If the dump is to print at 8 lines per inch, code FCB=STD3.

Examples of the SYSABEND and SYSUDUMP DD Statements

```
//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.

```
//STEP3 EXEC PGM=B
//SYSUDUMP DD SYSOUT=F
```

The SYSUDUMP DD statement specifies that you want the dump routed through the output class F.

```
//STEP1 EXEC PGM=PROGRAM1
//SYSABEND DD DSNAME=DUMP,UNIT=2314,
// DISP=(,PASS,KEEP),
// VOLUME=SER=1234,SPACE=(TRK,110)
//STEP2 EXEC PGM=PROGRAM2
//SYSABEND DD DSNAME=*.STEP1.SYSABEND,
// DISP=(OLD,DELETE,KEEP)
```

The SYSABEND DD statement specifies that you want the dump stored. The space request in STEP1 is large (110 tracks) so the dumping operation is not inhibited due to insufficient space. If STEP1 does not ABEND but STEP2 does, the dump is 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 ABENDS. If both of the steps are successfully executed, the second subparameter of the DISP parameter (DELETE) in STEP2 causes the data set to be deleted and the space acquired for dumping to be freed.

```
//STEP1 EXEC PGM=WWK
//SYSUDUMP DD DSNAME=DUMP,UNIT=2314,DISP=(,DELETE,
// KEEP),VOLUME=SER=54366
//STEP2 EXEC PGM=PRINT,COND=ONLY
//IN DD DSNAME=*.STEP1.SYSUDUMP,DISP=(OLD,
// DELETE),VOLUME=REF=*.STEP1.SYSUDUMP
```

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

```
//STEP5 EXEC PGM=PRINTS
//SYSABEND DD SYSOUT=A,CHARS=DUMP
```

The SYSABEND DD statement states that if an abend occurs, the dump is to be routed through the output class A. The specification of CHARS=DUMP causes the dump, if printed on a 3800, to print in the high-density format.

To obtain the high-density dump at 8 lines per inch, code FCB=STD3 as follows:

```
//SYSABEND DD SYSOUT=A,CHARS=DUMP,FCB=STD3
```

SYSCHK Facility

DD Statement

The SYSCHK DD statement defines a checkpoint data set written during the original execution of a processing program.

For detailed information about the checkpoint/restart facilities, see *OS/VS1 Checkpoint/Restart*, GC26-3876.

Rules for Coding

General

1. 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.)
2. Include a SYSCHK DD statement among the DD statements for a job whenever a deferred checkpoint restart is to occur, that is, whenever a job is resubmitted for restart of execution at a particular checkpoint.
3. If you include a JOBLIB DD statement, the SYSCHK DD statement must follow it.
4. Code the RESTART parameter on the JOB statement; otherwise the SYSCHK DD statement is ignored.
5. The rules for coding parameters on the SYSCHK DD statement depend on whether the checkpoint data set is cataloged. These rules are discussed under separate headings.

When the Checkpoint Data Set is Cataloged

1. Code the DSNNAME parameter, specifying the name of the checkpoint data set.
2. Code the DISP parameter, specifying or implying a status of OLD and a disposition of KEEP.
3. If the checkpoint entry exists on a tape volume other than the first volume of the checkpoint data set, code the VOLUME parameter, specifying either the volume sequence number or the volume serial number. (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.)
4. If you code the volume serial number, code the UNIT parameter as well.
5. If the checkpoint data set does not have standard labels, code the LABEL parameter.

6. If the checkpoint data set is on 7-track magnetic tape with nonstandard labels or no labels, code DCB=TRTCH=C.

When the Checkpoint Data Set is not Cataloged

1. Code the DSNNAME parameter, specifying the name of the checkpoint data set. If the checkpoint data set is partitioned, do not include a member name in the DSNNAME parameter.
2. Code the DISP parameter, specifying or implying a status of OLD, and a disposition of KEEP.
3. Code the VOLUME parameter, specifying 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.)
4. Code the UNIT parameter, specifying the device to be allocated to the data set.
5. If the checkpoint data set does not have standard labels, code the LABEL parameter.
6. If the checkpoint data set is on 7-track magnetic tape with nonstandard or no labels, code DCB=TRTCH=C.

Examples of the SYSCHK DD Statement

```
//JOB1      JOB      RESTART=( STEP3,CK3 )
//SYSCHK    DD        DSNNAME=CHLIB,UNIT=2314,
//          //        DISP=OLD,VOLUME=SER=456789
//STEP1     EXEC
```

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

```
//JOB2      JOB      RESTART=( STEP2,NOTE2 )
//JOBLIB    DD        DSNNAME=PRIV.LIB3,DISP=( OLD,PASS )
//SYSCHK    DD        DSNNAME=CHECKPTS,DISP=( OLD,KEEP ),
//          //        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.

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

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

* Parameter

Optional

The * parameter specifies that data follows the DD statement. When the * parameter is used, the system checks for an input delimiter (/ * or // or EOF) on the input reader device.

```
//ddname      DD *
```

Rules for Coding

1. You can code more than one DD * statement per job step.
2. If you specify a program name on the EXEC statement, you can include the data for the step in the input stream.
3. If you call a cataloged procedure with the EXEC statement, you can include the data for each procedure step in the input stream. You can add more than one DD * statement to each procedure step.
4. A cataloged procedure cannot contain a DD * statement.
5. You can use the DSID parameter and the VOL=SER parameter to define to the system reader a diskette data set that is to be read as associated data for this DD statement.

6. The keywords allowed on the DD * statement are: DLM, DCB=BLKSIZE, DCB=BUFNO, DCB=LRECL, VOLUME, DSID.

All other keywords found on the DD * statement cause a JCL error.

The VOL=SER parameter and DSID parameter are interpreted, but are used only when they are detected by the system reader as a request for a diskette data set.

Do not code symbolic parameters in the DSID, DLM, or VOLUME keywords on a DD * statement.

7. Code the DATA parameter instead of the * parameter when the data contains statements starting with //.
8. When you precede the data with a DD * statement, a delimiter statement (/ *) following the data is optional.
9. You may include input stream data on any device supported by QSAM, for example, a card reader, a magnetic tape, or a direct access device.
10. You must code input stream data records in BCD or EBCDIC.
11. You can include several distinct groups of data in the input stream for a job step or a procedure step. The system recognizes each group of data if you precede each group with a DD * statement, or if you follow each group with a delimiter statement (/ *), or both. (If you leave out the DD * statement, the system provides a DD * statement having SYSIN as its ddname. If more than one SYSIN is generated by the system for a job, only the first will be recognized.)

12. If the processing program does not read all the data in an input stream, the remaining data is flushed without causing an ABEND.
13. When a job is submitted via remote job entry and the DCB subparameter BUFNO is coded on a DD * statement, BUFNO is ignored.
14. You cannot use a backward reference to a previously defined DD statement to obtain the BLKSIZE and BUFNO subparameters. Code these DCB subparameters either on the DD * statement or on a DD statement containing the DDNAME parameter that refers to another DD statement.

If the referenced DD statement contains its own values for BLKSIZE and BUFNO, these values override the subparameters on the DD statement containing the DDNAME parameter.

Examples of the * Parameter

```
//INPUT1          DD      *
                  .
                  data
                  .
/*
//INPUT2          DD      *,DCB=(BLKSIZE=1600,BUFNO=2)
                  .
                  data
                  .
/*
```

Defines groups of data in the input stream. The DCB subparameters BLKSIZE and BUFNO override those specified in the input reader procedure.

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

Defines data in the input stream. 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.

DATA Parameter

Optional

The DATA parameter specifies that the data following the DD statement is to be entered through the input stream for use by a processing program. This data may contain statements with the characters // in columns 1 and 2.

```
//ddname      DD      DATA
```

Rules for Coding

1. You may code more than one DD DATA statement per job step.
2. If you had an EXEC statement for the job step that calls a cataloged procedure, you can add more than one DD DATA statement to a procedure step.
3. A cataloged procedure cannot contain a DD DATA statement.
4. You can use the DSID parameter and the VOL=SER parameter to define to the system reader a diskette data set that is to be read as associated data for this DD statement.
5. The keywords allowed on the DD DATA statement are: DLM, DCB=BLKSIZE, DCB=BUFNO, DCB=LRECL, VOLUME and DSID.
All other keywords found on the DD DATA statement cause a JCL error.
The VOL=SER parameter and DSID parameter are interpreted, but are used only when they are detected by the system reader as a request for a diskette data set.
Do not code symbolic parameters in the DSID, DLM, or VOLUME keywords on a DD DATA statement.
6. Code the * parameter instead of the DATA parameter when the data does not contain statements starting with //.
7. You can include several distinct groups of data in the input stream for a job step or a procedure step. Precede each group of data with a DD DATA statement and follow it with a delimiter statement (/*). The data contained between the DD DATA statement and the delimiter (/*) must not contain /* in columns 1 and 2.
8. You can place input stream data on any device supported by QSAM, for example, a card reader, a magnetic tape, or a direct access device.
9. You must code input stream data records in BCD or EBCDIC.
10. If the processing program does not read all the data in an input stream, the remaining data is flushed without causing an ABEND.
11. When a job is submitted via remote job entry and the DCB subparameter BUFNO is coded on a DD statement, BUFNO is ignored.
12. Do not use a backward reference to a previously defined DD statement to obtain the BLKSIZE and BUFNO subparameters. Code these DCB subparameters either on the DD DATA statement or on a DD

statement containing the DDNAME parameter that refers to another DD statement.

If the referenced DD statement contains its own values for BLKSIZE and BUFNO, these values override the subparameters on the DD statement containing the DDNAME parameter.

Examples of the DD DATA Parameter

```
//INPUT1          DD      DATA
                  .
                  data
                  .
/*
```

Defines data in the input stream.

```
//STEP2          EXEC    PROC=UPDATE
//PREP.DD4       DD      DSNNAME=A.B.C,
//              VOLUME=SER=D88,
//              UNIT=2314,SPACE=(TRK,(10,
//              5)),DISP=(,CATLG,DELETE)
//PREP.INPUT     DD      DATA
                  .
                  data
                  .
/*
//ADD.IN         DD      *
                  .
                  data
                  .
/*
```

Defines 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. Because this data is defined by a DD * statement, it must not contain job control statements.

```
//INPUT2         DD      DATA,DCB=(BLKSIZE=400,
//              BUFNO=1)
                  .
                  data
                  .
/*
//INPUT3        DD      DATA
                  .
                  data
                  .
/*
```

Defines groups of data in the input stream. The DCB subparameter coded on the DD statement named INPUT2 will be used to block the data that follows that statement.

DUMMY Parameter

Optional

The DUMMY parameter 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.
- For BSAM and QSAM, no input or output operations are to be performed on the data set.

For further information on the DUMMY parameter, see *OS/VSI JCL Services*, GC24-5100.

```
//ddname DD DUMMY
```

Rules for Coding

1. Code the DUMMY parameter by itself or follow it with all the parameters you would normally code when defining a data set, except the DDNAME parameter. The DDNAME and DUMMY parameters are mutually exclusive.
2. If you used the DUMMY parameter to test a program, 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.
3. When you want to nullify a procedure DD statement that contains the DUMMY parameter, code the DSNAME parameter on the overriding DD statement. However, be sure that the data set name is not NULLFILE, as it has the same effect as coding DUMMY.
4. If you code the DUMMY parameter and also request an access method other than BSAM (basic sequential access method) or QSAM (queued sequential access method) to read or write the data set, a programming error occurs.
5. When the DUMMY parameter is coded, all other parameters on the DD statement, with the exception of the DCB parameter, are checked for syntax and ignored. The DCB parameter must be coded if you would code it for normal I/O operations.
6. *Backward references:* 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

```
//OUTPUT3      DD  DUMMY,DSNAME=X.X.Z,UNIT=2314,  
//              SPACE=(TRK,(10,2)),DISP=(,CATLG)
```

This DD statement defines a dummy data set. The parameters coded with the DUMMY parameter will not be used.

```
//IN           DD  DUMMY,DCB=(BLKSIZE=800,  
//              LRECL=400,RECFM=FB)
```

This DD statement defines a dummy data set. The DCB parameter supplies information that was not supplied in the DCB macro instruction for the DCB. Otherwise, an ABEND may occur.

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
```

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
```

AFF Parameter

Keyword, Optional

The AFF parameter requests channel separation for a job step. When you use two or more data sets, processing time may be shortened if the system transmits data over separate channels.

AFF=ddname

ddname

the name of 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 by this DD statement to have the same channel separation as the data set defined on the named DD statement.

Rules for Coding

1. The DD statement that the AFF parameter refers to must contain the SEP parameter. If the SEP parameter is not present on that DD statement, channel separation does not occur.
2. The AFF, SEP, DDNAME, and SYSOUT parameters are mutually exclusive parameters; therefore, when you code SEP, DDNAME, or SYSOUT, do not use the AFF parameter.
3. 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 *UNIT Parameter*.
4. The AFF parameter does not tell the system that the data set to be used and the one referred to are to be assigned to the same channel; the system decides that, based on what devices are available for allocation.
5. *Satisfying a request for channel separation:* If the system finds it impossible in the current environment to satisfy the request for channel separation, it may attempt to alter the current environment through operator intervention. The operator has the option of bringing a device online, canceling the request for channel separation, or canceling the job. In certain environments, the operator may also be able to tell the system to wait for a device to become free.
6. *Nonspecific request:* If you make a nonspecific request for a direct access volume and also 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 permits the desired channel separation.
7. Requests for channel separation are ignored for any data sets that have been allocated by the automatic volume recognition (AVR) option.

8. 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

```
//STEP1      EXEC      PGM=CONVERT
//INPUT1     DD        DSNNAME=A.B.C,DISP=OLD
//INPUT2     DD        DSNNAME=FILE,DISP=OLD,UNIT=2400,
//           VOLUME=SER=54333
//BUF        DD        UNIT=2400,SEP=( INPUT1,INPUT2 )
//OUTPUT     DD        DSNNAME=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.

AMP Parameter (VSAM only)

Keyword, Optional

The AMP parameter modifies the program processing attributes for a VSAM data set for the duration of the job step.

For additional information about the AMP parameter, see *OS/VS Virtual Storage Access Method (VSAM) Programmer's Guide, GC26-3838*.

```
AMP=  ['AMORG']  
      ['BUFND=number']  
      ['BUFNI=number']  
      ['BUFSP=number']  
      [ 'CROPS=  { RCK' }  
        { NCK' }  
        { NRE' }  
        { NRC' } ]  
      [ 'OPTCD=  { P' }  
        { L' }  
        { IL' } ]  
      [ 'RECFM=  { F' }  
        { FB' }  
        { V' }  
        { VB' } ]  
      ['STRNO=number']  
      ['SYNAD=modulename']  
      ['TRACE']
```

AMORG

tells the operating system that the DD statement identifies a VSAM data set. AMORG must be specified when unit and volume information (or DUMMY) is included in the DD statement.

BUFND

specifies the number of I/O buffers to be used for transmitting the contents of data control intervals between virtual and auxiliary storage. A minimum of two data buffers is required. If the number of buffers is not specified in the AMP parameter or the ACB (or GENCB) macro, the default is one data buffer for each RPL string, plus one additional buffer.

BUFNI

specifies the number of I/O buffers to be used for transmitting the contents of index control intervals between virtual and auxiliary storage. A minimum of one index buffer is required. If the number of index buffers is not specified in the AMP parameter or ACB (or GENCB) macro, the default is one index buffer per RPL string. If the ISAM interface program is being used, a search of the high-level index in virtual storage can be simulated by adding one additional index buffer.

BUFSP

specifies the size of the user area to be allocated for data and index buffers. With entry-sequenced data sets, the minimum number of buffers required is two; with key-sequenced data sets, the minimum number of buffers required is three. If BUFSP is not specified, a space large enough to hold one data buffer and one index buffer for each RPL string is obtained, plus one additional buffer. If a control interval split is expected, an area large enough to hold all data and index buffers should be specified. If you specify less space than was specified in the BUFFERSPACE parameter of the DEFINE command of Access Method Services when the data set was defined, VSAM uses the BUFFERSPACE amount.

BUFSP has precedence over BUFND and BUFNI.

CROPS=RCK, NCK, NRE, or NRC

specifies the options applied to a VSAM component or cluster restarting at checkpoint.

RCK

specifies that data-erase and data-set-post-checkpoint modification tests are to be performed.

NCK

specifies that the data-set-post-checkpoint modification test is not to be performed.

NRE

specifies that the data-erase test is not to be performed.

NRC

specifies that neither data-erase nor data-set-post-checkpoint modification is to be performed.

For additional information, see *OS/VS1 Checkpoint/Restart*, GC26-3876.

OPTCD=L, I, or IL

specifies how the ISAM interface program is to handle records flagged for deletion.

L

indicates that a record marked for deletion (binary 1s in the first byte) is actually deleted only if it is replaced by a record with the same key. If your program writes a record marked for deletion, the ISAM interface puts it into the VSAM data set. This parameter is not required if you code OPTCD=L in the DCB in the processing program.

I

indicates the ISAM interface supplements normal ISAM processing for records marked for deletion. If OPTCD=I is specified but OPTCD=L is not specified in the processing program's DCB, records flagged for deletion are treated as any other records.

IL

indicates that if your processing program writes a record marked for deletion, the ISAM interface does not put the record into the data set. AMP=OPTCD=IL overrides OPTCD=L in the DCB in the processing program.

RECFM=F, FB, V, or VB

indicates to the ISAM interface program the ISAM record format for which your processing program is coded.

F

indicates fixed-length records.

FB

indicates blocked fixed-length records.

V

indicates variable-length records.

VB

indicates blocked variable-length records.

STRNO

specifies the number of concurrent RPL strings.

SYNAD

specifies the name of the module containing the user's SYNAD routine. If specified when using the ISAM interface program, the module name is interpreted as an ISAM SYNAD routine. This operand overrides the SYNAD specification in the EXLST macro.

TRACE

enables you to use the GTF (Generalized Trace Facility) to gather information about opening and closing data sets and end-of volume processing.

Rules for Coding

1. If the number of buffers specified in the BUFND and BUFNI subparameters causes the virtual storage requirements to exceed the BUFSP specification, the number of buffers is reduced to comply with BUFSP. If BUFSP specifies more space than required by BUFND and BUFNI, the number of buffers is increased.

For a key-sequenced data set, the total minimum buffer requirement for a single RPL string is three: two data buffers and one index buffer. For an entry-sequenced data set, two data buffers are required.

2. If only one subparameter is coded, do not code the leading comma.
3. Apostrophes must enclose each subparameter or group of subparameters, if they contain special characters; for example, AMP='BUFSP=value'. If the subparameters continue from one line to another, each line of subparameters must begin and end with an

apostrophe and the entire group of subparameters must be enclosed in parentheses. Reference the following examples.

Examples of the AMP Parameter

```
//AMPDD      DD      DSN=SYS1.MACLIB,DISP=SHR,  
//              AMP=( 'BUFSP=200,BUFND=2',  
//                    'BUFNI=3,STRNO=4,SYNAD=ERROR' )
```

This DD statement defines the size of the user area for data and index buffers; specifies the number of data and index buffers; specifies the number of concurrent RPL strings; and specifies an error analysis routine, ERROR, to override the error analysis routine specified in the EXLST macro.

```
//AMPDD      DD      DSN=SYS1.MACLIB,DISP=SHR,  
//              AMP=( 'BUFSP=23456,BUFND=5,BUFNI=10',  
//                    'STRNO=6,SYNAD=ERROR2',  
//                    'CROPS=NCK,TRACE' )
```

This DD statement defines the values for BUFSP, BUFND, BUFNI, STRNO, and SYNAD as in the previous example. It also specifies that a data-set-post-checkpoint modification test is not to be performed when restarting at a checkpoint and that OPEN is to provide a module trace.

Another way of continuing subparameters from one line to another is:

```
//AMPDD      DD      DSN=SYS1.MACLIB,DISP=SHR,  
//              AMP=( 'BUFSP=23456','BUFND=5',  
//                    'BUFNI=10','STRNO=6','SYNAD=ERROR2',  
//                    'RECFM=VB','CROPS=NCK','TRACE' )
```

BURST Parameter

Keyword, Optional

The BURST parameter is used to specify whether the paper output is to be in continuous forms or to go to the Burster-Trimmer-Stacker on the 3800.

For further information on the BURST parameter, see *IBM 3800 Printing Subsystem Programmer's Guide*, GC26-3846.

BURST={Y}
{N}

Y

indicates that the printed output is to be burst into separate sheets.

N

indicates that the printed output is to be continuous fanfold.

Rules for Coding

1. For SYSOUT processing, if BURST is not specified, JES will supply the default.
2. If UNIT is coded but BURST is not, the default is continuous forms.
3. Refer to Figure 9 for parameters that are mutually exclusive with BURST.

Examples of the BURST Parameter

//DD1 DD SYSOUT=A,BURST=Y

Requests that the output be burst into separate sheets.

//DD2 DD UNIT=018,BURST=N

Requests that the output remain on continuous forms.

CHARS Parameter

Keyword, Optional

The CHARS parameter identifies the character arrangements to be used in printing a data set on a 3800 Printing Subsystem.

For further information of the definition, selection and processing of the character arrangements, see the *IBM 3800 Printing Subsystem Programmer's Guide*, GC26-3846.

CHARS=(table name,...)

table name

a one-to-four character name of a character arrangement table module that resides in SYS1.IMAGELIB. As many as four names may be specified. Figure 13 gives the names of the character arrangement tables supplied with the 3800.

Rules for Coding

1. The subparameter cannot be null.
2. If only one table name is coded, you do not need the parentheses.
3. Refer to Figure 9 for parameters that are mutually exclusive with CHARS.
4. For SYSOUT processing, if you specify both CHARS and UCS, the JES writer uses CHARS for the 3800 and UCS for 3211 printers.
5. For SYSOUT processing, if you specify only UCS and the output device is a 3800, the JES writer uses the UCS parameter to set up the printer.
6. If the 3800 is allocated directly to the problem program, the UCS is ignored.
7. For SYSOUT processing, if you don't specify UCS or CHARS and the output device is a 3800, the value specified in the writer procedure is used. If no value is in the writer procedure, the hardware default Gothic-10 folded character arrangement table is used.

Examples of the CHARS parameter

```
//DD1      DD      SYSOUT=A,CHARS=(GS15,2773)
```

The DD1 statement requests the character arrangement tables GS15 and 2773 to be used when printing the data set on the 3800.

```
//DD2      DD      SYSOUT=A,CHARS=GS10,UCS=PN
```

The DD2 statement specifies that if the output device is a 3800, the GS10 character arrangement table will be used. If the device is a 3211 printer, the UCS image PN will be used.

```
//DD3      DD      UNIT=3800,CHARS=T11
```

The DD3 statement requests the character arrangement table T11 for the directly allocated 3800.

CHKPT Parameter

Keyword, Optional

Use the CHKPT parameter to invoke the checkpoint at EOVS facility. It specifies that checkpoints are to be taken for the data set defined by the DD statement on which it is coded. For more information see *OS/VS1 Checkpoint Restart*, GC26-3876.

CHKPT=EOV

EOV

specifies that checkpoints occur at end-of-volume. This is the only allowable value for this parameter.

Rules for Coding

1. Specify CHKPT=EOV parameter only for multivolume data sets using QSAM or BSAM.
2. You can code either input or output data sets.
3. This parameter is mutually exclusive with DD *, DD DATA, SYSOUT, and DDNAME parameters. Coding CHKPT=EOV with any of these results in a JCL error.
4. Specifying CHKPT for non-multivolume BSAM/QSAM data sets or for ISAM, BDAM, BPAM, or VSAM data sets does not result in any checkpoint processing. Processing continues as if the parameter was not coded at all.
5. You must code this parameter on each DD statement for concatenated BSAM or QSAM data sets if you want checkpointing for each DD statement.
6. If you specify this parameter on one or more DD statements in a job step, you must provide a SYSCKEOV DD statement as outlined in *OS/VS1 Checkpoint Restart*.
7. Use the RD parameter values NC and RNC on the JOB or EXEC statements to suppress the action of the CHKPT=EOV parameter.
8. The CHKPT parameter is suppressed for started tasks if specified within the procedure for the started task, or in the START console command.

Examples of the CHKPT Parameter

```
//DS1      DD      DSNAME=INDS,DISP=OLD,UNIT=TAPE,  
//          VOL=SER=(TAPE01,TAPE02,TAPE03),  
//          CHKPT=EOV
```

INDS is a multivolume QSAM (or BSAM) data set for which a checkpoint is to be taken twice; once after end-of-volume on TAPE01 and once after end-of-volume on TAPE02.

```
//DS2      DD      DSNAME=OUTDS,DISP=(NEW,KEEP),  
//          UNIT=DISK,VOL=( , , , 8 ),CHKPT=EOV
```

OUTDS is a multivolume data set that requires eight volumes. Seven checkpoints will be taken; one after each of the first seven volumes.

COMPACT Parameter

Keyword, Optional

The COMPACT parameter is specified on the SYSOUT DD statement for a 3790 and identifies the Compaction Table. For more information, see *OS/VS1 RES System Programmer's Guide*, GC28-6878.

$$\text{COMPACT} = \left\{ \begin{array}{l} \text{NO} \\ \text{compact table id} \end{array} \right\}$$

NO

indicates that compaction will not occur.

compact table id

is the name of the Compaction Table. It can be a maximum of four characters long. It can contain alphabetic, numeric, and national characters. The first character must be alphabetic or national.

Rules for Coding

1. If coded on other than a SYSOUT DD statement, COMPACT is syntax checked, but otherwise ignored.
2. If RES is not available, COMPACT has no meaning. It is syntax checked, but otherwise ignored.
3. COMPACT is mutually exclusive with DD *, DD DATA, DDNAME, and itself.

Examples of the COMPACT Parameter

```
//OUT1 DD SYSOUT=A,COMPACT=NO
```

Indicates that when the data set is sent to a remote workstation, it will not be compacted.

```
//OUT2 DD SYSOUT=A,COMPACT=LIST
```

Indicates that when the data set is sent to a remote workstation, the Compaction Table LIST be used.

COPIES Parameter

Keyword, Optional

The COPIES parameter specifies the number of copies to be printed for a data set, and, if printing is done on a 3800, how the copies are to be grouped.

For further information on the use of the COPIES parameter, see *OV/VSI JCL Services*, GC24-5100. Additional information on COPIES using the 3800 can be found in the *IBM 3800 Printing Subsystem Programmer's Guide*, GC26-3846.

COPIES=(nnn[(group value,...)])

nnn

specifies the total number of copies of a data set to be printed. If printing on a 3800 and you specify the additional subparameter group values, the total of the group values, instead of nnn, is used as the total number of copies.

group value

is a subparameter for the 3800 only and specifies the way in which output is to be grouped. Each group value may be from 1 to 255 with the total of all group values not exceeding 255. The maximum number of group values is 8. Each group value represents the number of consecutive copies of a page to be printed (for example, a group value of 3 causes the first page of a data set to be printed three times before printing is started for the second page). When you specify group values together with the subparameter nnn, the group values are used to determine the number of copies, not exceeding 255, to be printed. nnn is used only when printing is reassigned from a 3800 to an impact printer.

Rules for Coding

1. You can code the COPIES parameter together with either the SYSOUT or UNIT parameters. You normally use the COPIES parameter with the SYSOUT parameter. The JES writer reprints the data set to satisfy the COPIES requirement. If, however, you use COPIES with the UNIT parameter, the subparameter nnn is defaulted to 1, and if printing on a 3800 and the subparameter group value is specified, the system sets up the 3800 to print the number of copies based on the first group value.
2. A null group value is invalid; for example, COPIES.
3. A zero or null position in the group value subparameter is invalid.
4. If only nnn is coded, you do not need parentheses.
5. Refer to Figure 9 for parameters that are mutually exclusive with COPIES.

Examples of the COPIES parameter

```
//DD1      DD      SYSOUT=A,COPIES=32
```

Thirty-two copies of the data set defined by the DD1 DD statement will be printed.

```
//DD2      DD      SYSOUT=A,COPIES=(0,(1,2))
```

This statement says an impact printer will print one copy (default), and if a 3800 is used instead, it will print three copies of the data set in two groups. The first group will have one copy of each page. The second group will have two copies of each page.

```
//DD3      DD      SYSOUT=A,COPIES=(8,(1,3,2))
```

This states that if the output device is a 3800, six copies of the data set print. Three groups will be produced. The first contains one copy of each page, the second contains three copies of each page, and the last contains two copies of each page. If the output device is not a 3800, eight separate copies will print.

```
//DD4      DD      UNIT=3800,COPIES=(1,(2,3))
```

This statement says the 3800 will print two copies of each page, because of the UNIT parameter.

```
//DD5      DD      UNIT=3211,COPIES=(1,(2,3))
```

This statement says the 3211 will print one copy of the data set.

DCB Parameter

Keyword, Optional

The DCB parameter is used to complete information in a DCB about a data set at execution time. The DCB is originally constructed in a processing program by a DCB macro instruction.

For further information on the formation of the DCB, see *OS/VS1 Data Management Services Guide*, GC26-3874.

VSAM control blocks are not created by the DCB macro or DCB parameter. They are created by control block definition macros described in *OS/VS Virtual Storage Access Method (VSAM) Programmer's Guide*, GC26-3838.

$$\text{DCB} = \left(\begin{array}{l} \text{(list of attributes)} \\ \left(\begin{array}{l} \text{dsname} \\ \text{*ddname} \\ \text{*stepname.ddname} \\ \text{*stepname.procstepname.ddname} \end{array} \right) \end{array} \right) \text{[,list of attributes]}$$

list of attributes

those DCB keyword subparameters that describe the data set and are needed to complete the DCB. The DCB keyword subparameters are listed alphabetically in this section according to access methods, for example, BDAM, BISAM.

dsname

the name of a cataloged data set from which the system is to copy DCB information. The information is contained in the data set label of the cataloged data set. The data set must reside on a direct access volume and the volume must be mounted before execution of the job step.

*.ddname

the name of an earlier DD statement in the same job step from which the system is to copy DCB information.

*.stepname.ddname

the name of a DD statement (ddname) in an earlier job step (stepname) from which the system is to copy DCB information.

*.stepname.procstepname.ddname

the name of a DD statement (ddname), which appears in a procedure step, (procstepname); the procedure step is part of a cataloged procedure that was called by an earlier job step (stepname).

Rules for Coding

General

1. Separate DCB keyword subparameters by a comma.
2. You need not enclose the DCB parameter value in parentheses if it consists of only one keyword subparameter, a data set name, or a backward reference.

3. All DCB subparameters, except BLKSIZE, BUFNO, and DIAGNS, are mutually exclusive with the DDNAME parameter; therefore, when the DDNAME parameter is coded, do not code any DCB subparameters except BLKSIZE, BUFNO, and DIAGNS.
4. Code the DCB parameter on the DD statement unless the DCB is completed by another source, for example, the DCB macro instruction in the processing program. You can specify the DCB information on the DD statement in several ways. The following methods are explained in detail in the next three groups of syntax rules:
 - Supplying all pertinent DCB keyword subparameters on the DD statement.
 - Copying the DCB information from the data set label of an existing cataloged data set.
 - Copying the DCB information from an earlier DD statement.

DCB Keyword Subparameters

1. You must code the DCB macro instruction in a processing program. However, you can supply some DCB operands as DCB subparameters on a DD statement.
2. List the information required to complete the DCB as keyword subparameters in the DCB parameter.
3. If the processing program and the DCB parameter supply the same subparameter, the subparameter on the DD statement is ignored.
4. The DCB keyword subparameters are listed alphabetically in this section according to access methods, for example, BDAM, BISAM.

DCB Information from a Data Set Label

1. You can copy DCB information from the data set label of a cataloged data set on a currently mounted direct access volume, or from any existing data that is opened for input. A permanently resident volume is the best place from which to copy information because it is always mounted.
2. Code in the DCB parameter the data set name of the cataloged data set. The data set name cannot contain special characters, except for periods used in a qualified name.
3. These DCB keyword subparameters can be copied from the data set label:

DSORG (used in a backward reference)
 RECFM
 OPTCD
 BLKSIZE
 LRECL
 KEYLEN
 RKP

The volume sequence number, system code, creation date, and expiration date of the cataloged data set are also copied unless you specify them in the DD statement.

4. 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.
5. The DCB subparameters are listed alphabetically in this section according to access methods, for example, BDAM, BISAM.

DCB Information from an Earlier DD Statement

1. The earlier DD statement from which DCB information can be copied can be contained in the same job step, an earlier job step, or a cataloged procedure step. Code in the DCB parameter one of the following types of reference names, depending on the location of the DD statement you want to use:

```
*.ddname
*.stepname.ddname
*.stepname.procstepname.ddname
```

2. 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.

The system copies only those subparameters from the earlier DD statement that are not again specified on the referencing DD statement.

3. The DCB subparameters are listed alphabetically in this section according to access methods, for example, BDAM, BISAM.

Examples of the DCB Parameter

```
//DD1 DD DSNAME=ALP,DISP=(,KEEP),VOLUME=SER=44321,
// UNIT=2400,DCB=(RECFM=FB,LRECL=240,
// BLKSIZE=960,DEN=1,TRTCH=C)
```

This DD statement defines a new data set and contains the information necessary to complete the DCB.

```
//DD2 DD DSNAME=BAL,DISP=OLD,DCB=(RECFM=F,LRECL=80,
// BLKSIZE=80)
//DD3 DD DSNAME=CNANN,DISP=(,CATLG,DELETE),UNIT=2400,
// 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.

```
//DD4 DD DSNAME=JST,DISP=(NEW,KEEP),UNIT=2314,
// 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 because KEYLEN is coded on the DD statement.


```
//DD5 DD DSNAME=SAME,DISP=OLD,UNIT=2314,  
// 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, five buffers are assigned to the DCB.

DCB Subparameters for BDAM

You can use these DCB subparameters with BDAM:

BFALN	BUFNO	LIMCT
BFTEK	DIAGNS	OPTCD
BLKSIZE	DSORG	RECFM
BUFL	KEYLEN	

The definitions and rules for coding these parameters follow.

BFALN Subparameter

The BFALN subparameter specifies the boundary of each buffer.

$$\text{BFALN} = \left\{ \begin{array}{l} \text{F} \\ \text{D} \end{array} \right\}$$

F

indicates that each buffer starts on a fullword boundary that is not also a doubleword boundary.

D

indicates that each buffer starts on a doubleword.

Default: If you do not specify BFALN, doubleword boundary alignment is assumed. If both the BFALN and BFTEK subparameters are specified, they must be supplied from the same source; that is, either through the JCL DCB parameter or through the DCB macro instruction.

BFTEK Subparameter

The BFTEK subparameter specifies that the data set is being created for or contains variable-length spanned records. The BFTEK subparameter can only be coded when the record format is specified as RECFM=VS.

$$\text{BFTEK} = \text{R}$$

If both the BFTEK and BFALN subparameters are specified, they must be supplied from the same source; that is, either through the JCL DCB parameter or through the DCB macro instruction.

BLKSIZE Subparameter

The BLKSIZE subparameter specifies the maximum length, in bytes, of a block. The largest number allowed is 32,760.

$$\text{BLKSIZE} = \text{number of bytes}$$

Blocksize varies according to record format (specified by the RECFM subparameter).

- If RECFM=F, BLKSIZE must be logical record length.
- If RECFM=V, BLKSIZE must be (maximum block size + 4).

If you code the BLKSIZE subparameter in the DCB macro instruction or on a DD statement that defines an existing data set with standard labels, the subparameter overrides the blocksize specified in the label.

BUFL Subparameter

The BUFL subparameter specifies the length, in bytes, of each buffer in the buffer pool. The maximum length allowed is 32,760.

BUFL=number of bytes

The BUFL subparameter is required for BDAM only if dynamic buffering is specified in the MACRF subparameter of the DCB macro instruction.

BUFNO Subparameter

The BUFNO subparameter specifies how many buffers are to be assigned to the DCB; the maximum normally is 255, but may be less because of limits established when the system was generated.

BUFNO=number of buffers

Requirements for coding the BUFNO subparameter are:

Method of obtaining the buffer pool:	Requirement for indicating the number of buffers:
BUILD macro instruction	BUFNO subparameter must be specified.
GETPOOL macro instruction	Control Program uses the number specified in the GETPOOL macro instruction.
Dynamic buffering	Optional; if not specified, two buffers are obtained.
Automatically	Must be specified.

DIAGNS Subparameter

The DIAGNS subparameter specifies the OPEN/CLOSE/EOV trace option which gives a module-by-module trace of OPEN/CLOSE/EOV's work area and the DCB.

DIAGNS=TRACE

If the subparameter is not specified on the DD card, the option is not implemented. The GTF (Generalized Trace Facility), with the proper options specified, must be active in the system while the job that requested the trace is running. The options that must be specified for the GTF are MODE=EXT and TRACE=USR.

DSORG Subparameter

The DSORG subparameter specifies the organization of the data set and indicates whether the data set contains any location-dependent information that would make the data set unmovable.

DSORG=data set organization

You can code the following values with the DSORG Subparameter:

DA

direct organization data set.

DAU

direct organization data set that contains location dependent information.

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.

KEYLEN Subparameter

The KEYLEN subparameter specifies the length, in bytes, of the keys used in the data set. The largest number allowed is 255.

KEYLEN=number of bytes

The key length information can be supplied from the data set label for an existing data set. If a key length is not specified, no input or output requests that require a key may be issued.

LIMCT Subparameter

The LIMCT subparameter specifies how many blocks (if relative block addressing is used) or how many tracks (if relative track addressing is used) are to be searched for a free block or available space. This kind of search occurs only when the extended search option is specified (OPTCD=E).

LIMCT=number of blocks

If the number specified in the LIMCT subparameter equals or exceeds the number of blocks or tracks in the data set, the entire data set is searched.

The LIMCT subparameter is ignored if the extended search option is not specified.

OPTCD Subparameter

The OPTCD subparameter requests optional services from the control program.

$$\text{OPTCD} = \left\{ \begin{array}{l} \text{A} \\ \text{R} \quad [\text{E}] [\text{F}] [\text{W}] \end{array} \right\}$$

All optional services must be requested by one method; the characters may be coded in any order and when used in combination, no commas are permitted between characters.

You can code these values with the OPTCD subparameter:

A

indicates that the actual device addresses are to be specified in READ and WRITE macro instructions.

R

indicates that relative block addresses are to be specified in READ and WRITE macro instructions.

E

indicates that an extended search (more than one track) is to be performed for a block or available space. With this value, the LIMCT subparameter must also be coded. Do not code LIMCT=0; it will cause an ABEND when a READ or WRITE macro instruction is issued.

F

indicates that feedback may be requested in READ and WRITE macro instructions and the device address returned is to be in the same form as that presented to the control program.

W

requests a validity check for write operations on direct access devices.

RECFM Subparameter

The RECFM subparameter specifies the format and characteristics of the records in the data set. The format and characteristics must be completely described by one source.

```
RECFM=  {U}
        {V [S] }
        { [BS] }
        {F [T] }
```

These values can be used with the RECFM subparameter:

U

indicates that the records are of undefined length.

V

indicates that the records are of variable length.

VS

indicates that the records are of variable length and spanned.

VBS

indicates that the records are of variable length, blocked, and spanned, and the problem program must block and segment the records.

F

indicates that the records are of fixed length.

T

indicates that the records may be written onto overflow tracks if required.

Default: If you omit the RECFM subparameter, an undefined length record is assumed with no optional features provided.

DCB Subparameters for BISAM

You can use the following DCB subparameters with BISAM:

BFALN **BUFNO** **NCP**
BUFL **DIAGNS**
 DSORG

The definitions and rules for coding these subparameters follow.

BFALN Subparameter

The BFALN subparameter specifies the boundary of each buffer.

BFALN= { F }
 { D }

F

indicates that each buffer starts on a fullword boundary that is not also a doubleword boundary.

D

indicates that each buffer starts on a doubleword boundary.

Default: If you omit BFALN, doubleword boundary alignment is assumed.

BUFL Subparameter

The BUFL parameter specifies the length, in bytes, of each buffer in the buffer pool. The maximum length is 32,760 bytes.

BUFL=number of bytes

The BUFL subparameter is not required if the control program acquires buffers automatically or if dynamic buffering is specified. For this access method, dynamic buffering is specified in the MACRF subparameter of the DCB macro instruction.

BUFNO Subparameter

The BUFNO subparameter specifies the number of buffers to be assigned to the DCB; the maximum is normally 255, but may be less because of limits established when the system was generated.

BUFNO=number of buffers

Requirements for coding the BUFNO subparameter are:

Method of obtaining the buffer pool:

BUILD macro instruction
GETPOOL macro instruction

Dynamic buffering

Requirement for indicating the number of buffers:

BUFNO subparameter must be specified.
Control program uses the number specified in the GETPOOL macro instruction.
Optional: if not specified, two buffers are obtained.

DIAGNS Subparameter

The DIAGNS subparameter specifies the OPEN/CLOSE/EOV trace option which gives a module-by-module trace of OPEN/CLOSE/EOV's work area and the DCB.

DIAGNS=TRACE

If the subparameter is not specified on the DD card, the option is not implemented. The GTF (Generalized Trace Facility), with the proper options specified, must be active in the system while the job that requested the trace is running. The options that must be specified for the GTF are MODE=EXT and TRACE=USR.

DSORG Subparameter

The DSORG subparameter specifies the organization of the data set and indicates whether the data set contains any location-dependent information that would make the data set unmovable.

DSORG=data set organization

You can code this value with the DSORG subparameter:

IS
indexed sequential

The DSORG subparameter must be coded on the DD statement.

NCP Subparameter

The NCP subparameter specifies the maximum number of READ or WRITE macro instructions that can be issued before a CHECK macro instruction is issued.

NCP=number of macros

Default: If you omit the NCP subparameter, one is assumed.

The maximum number is normally 99, but may be less, depending on the amount of virtual storage available in the region or partition.

If dynamic buffering is used, the value specified for the NCP subparameter must not exceed the number of buffers specified in the BUFNO subparameter.

DCB Subparameters for BPAM

You can use these DCB subparameters with BPAM:

BFALN	DIAGNS	NCP
BLKSIZE	DSORG	OPTCD
BUFL	KEYLEN	RECFM
BUFNO	LRECL	

The definitions and rules for coding these subparameters follow.

BFALN Subparameter

The BFALN subparameter specifies the boundary of each buffer.

$$\text{BFALN} = \left\{ \begin{array}{l} \text{F} \\ \text{D} \end{array} \right\}$$

F

indicates that each buffer starts on a fullword boundary that is not also a doubleword boundary.

D

indicates that each buffer starts on a doubleword boundary.

Default: If you omit BFALN, doubleword boundary alignment is assumed.

BLKSIZE Subparameter

The BLKSIZE subparameter specifies the maximum length, in bytes, of a block. The maximum length is 32,760.

$$\text{BLKSIZE} = \text{number of bytes}$$

Blocksize varies according to record format (specified by the RECFM subparameter).

- If RECFM=F, BLKSIZE must be logical record length.
- If RECFM=FB, BLKSIZE must be an integral multiple of the logical record length.
- If RECFM=V, BLKSIZE must be (logical record length + 4).
- If RECFM=VB, BLKSIZE must be (n times logical record length) + 4, where n is the number of logical records in the block.

If you code the BLKSIZE subparameter in the DCB macro instruction or on a DD statement that defines an existing data set with standard labels, the subparameter overrides the blocksize specified in the label.

BUFL Subparameter

The BUFL subparameter specifies the length, in bytes, of each buffer in the buffer pool. The maximum length is 32,760 bytes.

$$\text{BUFL} = \text{number of bytes}$$

The BUFL subparameter is optional; if you omit it and the control

program acquires buffers automatically, the blocksize and key length information is used to establish the buffer length.

BUFNO Subparameter

The BUFNO subparameter specifies the number of buffers assigned to the DCB; the maximum is normally 255, but may be less than 255 because of limits established when the system was generated.

BUFNO=number of buffers

Requirements for coding the BUFNO subparameter are:

Method of obtaining the buffer pool:	Requirement for indicating the number of buffers:
BUILD macro instruction	BUFNO subparameter must be specified.
GETPOOL macro instruction	Control program uses the number specified in the GETPOOL macro instruction.
Automatically	Must be specified.

DIAGNS Subparameter

The DIAGNS subparameter specifies the OPEN/CLOSE/EOV trace option which gives a module-by-module trace of OPEN/CLOSE/EOV's work area and the DCB.

DIAGNS=TRACE

If the subparameter is not specified on the DD card, the option is not implemented. The GTF (Generalized Trace Facility), with the proper options specified, must be active in the system while the job that requested the trace is running. The options that must be specified for the GTF are MODE=EXT and TRACE=USR.

DSORG Subparameter

The DSORG subparameter specifies the organization of the data set and indicates whether the data set contains any location-dependent information that would make the data set unmovable. The DSORG subparameter must always be coded on the DCB macro instruction.

DSORG=data set organization

You can code these values with the DSORG subparameter:

PO

specifies a partitioned data set organization.

POU

specifies a partitioned data set organization and that the data set contains location dependent information.

KEYLEN Subparameter

The KEYLEN subparameter specifies the length, in bytes, of the keys used in the direct access device data set. The largest number allowed is 255.

KEYLEN=number of bytes

Default: If key length information is not supplied by any source before the OPEN macro instruction is issued, a length of zero (no keys) is assumed.

If standard labels are used, the key length information can be supplied from the data set label for an existing data set.

LRECL Subparameter

The LRECL subparameter specifies the actual or maximum length, in bytes, of each fixed length logical record in the data set.

LRECL=number of bytes

The record length is required for fixed-length records. The length cannot exceed the blocksize (BLKSIZE).

Blocksize varies according to record format (specified in the RECFM subparameter).

- If RECFM=V or VB, LRECL must be equal to the largest logical record length + 4.
- If RECFM=F or FB, LRECL must be equal to the largest logical record length.
- If RECFM=U, LRECL should be omitted.

The record length is required for fixed-length records only.

NCP Subparameter

The NCP subparameter specifies the maximum number of READ or WRITE macro instructions that can be issued before a CHECK macro instruction is issued.

NCP=number of macros

Default: If you omit NCP, one is assumed.

The largest number that can be specified in the NCP subparameter is 99, but may be less, depending on the amount of virtual storage available in the region or partition.

If chained scheduling is used, NCP must be specified as more than one.

OPTCD Subparameter

The OPTCD subparameter specifies the optional services to be performed by the control program.

OPTCD= {C|H|W|CH|CW|HW|CHW}

All optional services must be requested by one method. The characters may be coded in any order and when used in combination, no commas are permitted between characters.

These values can be used with the OPTCD subparameter:

C

requests that chained scheduling be used.

H

requests that if a Partitioned Data Set residing on a mass storage device (3850) is opened for input, the data set be staged to EOF on the virtual DASD. Otherwise, only the directory is staged.

W

requests a validity check for write operations on direct access devices.

RECFM Subparameter

The RECFM subparameter specifies the format and characteristics of the records in a data set. The format and characteristics must be completely described by one source.

$$\text{RECFM} = \left(\begin{array}{l} \text{U} \left[\begin{array}{l} \text{T} \\ \text{A} \\ \text{M} \end{array} \right] \\ \text{V} \left[\begin{array}{l} \text{B} \\ \text{T} \\ \text{BT} \end{array} \right] \left[\begin{array}{l} \text{A} \\ \text{M} \end{array} \right] \\ \text{F} \left[\begin{array}{l} \text{B} \\ \text{T} \\ \text{BT} \end{array} \right] \left[\begin{array}{l} \text{A} \\ \text{M} \end{array} \right] \end{array} \right)$$

These values can be used with the RECFM subparameter:

A

indicates that the record contains ASA control characters for ANSI printer and/or stacker selection.

B

indicates that the records are blocked.

F

indicates that the records are of fixed length.

M

indicates that the records contain machine code control characters.

T

indicates that the records may be written onto overflow tracks if required. Exchange buffering or chained scheduling (OPTCD=C) cannot be used.

U

indicates that the records are of undefined length.

V

indicates that the records are of variable length.

Default: If you omit the RECFM subparameter, an undefined-length record is assumed with no optional features provided.

DCB Subparameters for BSAM

You can use the following DCB subparameters with BSAM:

BFALN	CODE	FUNC	OPTCD
BFTEK	DEN	KEYLEN	PRTSP
BLKSIZE	DIAGNS	LRECL	RECFM
BUFL	DSORG	MODE	STACK
BUFNO	FRID	NCP	TRTCH
BUFOFF			

The definitions and rules for coding these subparameters follow.

BFALN Subparameter

The BFALN subparameter specifies the boundary of each buffer.

$$\text{BFALN} = \left\{ \begin{array}{l} \text{F} \\ \text{D} \end{array} \right\}$$

F

indicates that each buffer starts on a fullword boundary that is not also a doubleword boundary.

D

indicates that each buffer starts on a doubleword boundary.

Default: If you omit BFALN, doubleword boundary alignment is assumed. If both the BFALN and BFTEK subparameters are specified, they must be supplied from the same source; that is, either through the JCL DCB parameter or through the DCB macro instruction.

BFTEK Subparameter

The BFTEK subparameter specifies that BSAM is used to read unblocked variable-length spanned records with keys from a BDAM data set.

$$\text{BFTEK} = \text{R}$$

If both the BFTEK and BFALN subparameters are specified, they must be supplied from the same source; that is, either through the JCL DCB parameter or through the DCB macro instruction.

BLKSIZE Subparameter

The BLKSIZE subparameter specifies the maximum length, in bytes, of a block. For fixed-length, unblocked records, this subparameter specifies the record length. The maximum length is 32,760. For magnetic tape the minimum number is 18. For blocks of ASCII records on magnetic tape, the largest number that can be specified is 2,048 and the smallest number is 18.

$$\text{BLKSIZE} = \text{number of bytes}$$

Blocksize varies according to record format (specified by the RECFM subparameter).

- If RECFM=F, BLKSIZE must be logical record length.

- If RECFM=FB, BLKSIZE must be an integral multiple of the logical record length, if LRECL is specified.
- If RECFM=V, BLKSIZE must be (maximum logical record length + 4).
- If RECFM=VB, BLKSIZE must be at least four greater than the maximum logical record length.
- If RECFM=D or RECFM=DB, BLKSIZE must be (maximum record length + block prefix length).

If you code the BLKSIZE subparameter in the DCB macro instruction or on a DD statement that defines an existing data set with standard labels, the subparameter overrides the blocksize specified in the label.

BUFL Subparameter

The BUFL subparameter specifies the length, in bytes, of each buffer in the buffer pool when the buffer is acquired automatically. The maximum is 32,760 bytes.

BUFL=number of bytes

Default: The BUFL subparameter is optional; if you omit it and the control program acquires buffers automatically, the blocksize (BLKSIZE) and key length (KEYLEN) information is used to establish buffer length.

BUFNO Subparameter

The BUFNO subparameter specifies how many buffers are assigned to the DCB; the maximum is normally 255, but may be less than 255 because of the partition size.

BUFNO=number of buffers

Requirements for coding the BUFNO subparameter are:

Method of obtaining the buffer pool:	Requirement for indicating the number of buffers:
BUILD macro instruction	BUFNO subparameter must be specified.
GETPOOL macro instruction	Control program uses the number specified in the GETPOOL macro instruction.
Automatically	Must be specified.

BUFOFF Subparameter

The BUFOFF subparameter 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.

$$\text{BUFOFF} = \left\{ \begin{array}{l} n \\ L \end{array} \right\}$$

n

a number that indicates the length, in bytes, of the block prefix. For input, n may be any unsigned decimal number from 0 through 99.

L

indicates that the block prefix field is four bytes long and contains the block length. L may be specified only when record format (RECFM) is D.

CODE Subparameter

The CODE subparameter specifies the paper tape code in which the data is punched.

The subparameters CODE, KEYLEN, MODE, PRTSP, STACK, and TRTCH are mutually exclusive subparameters. Therefore, if you use CODE, do not use any of these other subparameters.

CODE= $\left. \begin{array}{c} A \\ B \\ C \\ F \\ I \\ N \\ T \end{array} \right\}$

A

ASCII (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).

Default: If you omit the CODE subparameter, I is assumed.

DEN Subparameter

The DEN subparameter specifies the magnetic tape density in number of bpi (bits per inch) used to write a data set.

DEN= $\left. \begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \right\}$

0

for 7-track tape, indicates 200 bpi.

1

for 7-track tape, indicates 556 bpi.

2

for 7-track tape and for 9-track tape, indicates 800 bpi.

3

for 9-track tape, indicates 1600 bpi.

4

for 9-track tape, indicates 6250 bpi.

Note: *When a tape is allocated to a 7-track drive with DEN=0 (200 bpi) for writing, the tape label and/or data may be written at 556 bpi instead. This occurs when a 3420 drive is allocated instead of a 2400 series or 3410. The 3420 executes the 200 bpi modeset as a 556 bpi modeset.*

Default: If you omit the DEN subparameter:

800 bpi is assumed for 7-track tape and for 9-track NRZI tape without dual density.

1600 bpi is assumed for 9-track tape with 1600/1800 dual density or phase-encoded tape.

6250 bpi is assumed for 9-track tape with 6250/1600 bpi dual density or group coded recording tape.

For 7-track tape, all information on the reel must be written in the same density, (that is, labels, data, tapemarks).

Do not specify DEN for a SYSOUT data set.

If not specified by any source, the highest applicable density is assumed.

DIAGNS Subparameter

DIAGNS subparameter specifies the OPEN/CLOSE/EOV trace option which gives a module-by-module trace of OPEN/CLOSE/EOV's work area and the DCB.

DIAGNS=TRACE

If the subparameter is not specified on the DD card, the option is not implemented. The GTF (Generalized Trace Facility), with the proper options specified, must be active in the system while the job that requested the trace is running. The options that should be specified for the GTF are MODE=EXT and TRACE=USR.

DSORG Subparameter

The DSORG subparameter specifies the organization of the data set and indicates whether the data set contains any location-dependent information that would make the data set unmovable.

The DSORG subparameter must always be coded on the DCB macro instruction.

DSORG=data set organization

You can code these values with the DSORG subparameter:

PS

indicates a physical sequential data set.

PSU

indicates a physical sequential data set that contains location dependent information.

FRID Subparameter

The FRID subparameter specifies an identifier for the first format record of the 3886 data set. A format record identifier must be specified for each 3886 data set.

FRID=identifier

The identifier can be one to four alphameric characters.

The FRID information can also be specified in the DCB macro instruction. If the DCB macro is used, the FRID information on the DCB macro overrides the FRID information on the DD statement.

FRID is mutually exclusive with the FCB parameter.

FUNC Subparameter

The FUNC subparameter specifies the type of data set to be opened for the 3525 Card Punch. If FUNC is omitted from all sources, a data set is opened for input defaults to read only and a data set is opened for output defaults to punch only.

FUNC=function

You can code these functions with the FUNC subparameter:

I

data in a data set is to be punched into and printed on cards.

R

data set is for reading cards.

P

data set is for punching cards.

W

data set is for printing.

D

data protection for a punch data set.

X

printer

T

two-line printer.

The only valid combinations of these values are:

I	WT	RWT	RPW
R	RP	PW	RPWXT
P	RPD	PWXT	RPWD
W	RW		

Note: If data protection is specified, the data protection image (DPI) must be specified in the FCB parameter.

KEYLEN Subparameter

The KEYLEN subparameter specifies the length, in bytes, of the keys used in the data set.

KEYLEN=number of bytes

Default: If you omit key length information, a length of zero (no keys) is assumed.

If standard labels are used, the key length information can be supplied from the data set label for an existing data set.

The subparameters KEYLEN, CODE, DEN, MODE, PRTSP, STACK, and TRTCH are mutually exclusive subparameters. Therefore, if KEYLEN is coded, do not code any of these other subparameters.

LRECL Subparameter

The LRECL subparameter specifies the actual or maximum length, in bytes, for fixed-length records or it specifies the maximum length, in bytes, for variable-length records.

LRECL= $\left\{ \begin{array}{l} n \\ x \end{array} \right\}$

n

number of bytes

x

specified for variable-length records that exceed 32,756.

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 blocksize (BLKSIZE) except for variable-length spanned records.

- If RECFM=V or VB, LRECL must be equal to the largest logical record length +4.
- If RECFM=F or FB, LRECL is the logical record length.
- If RECFM=U, LRECL should be omitted.
- If RECFM=D or DB, LRECL must be equal to the maximum record length.

You can omit the record length from all sources, in which case use the blocksize specification (BLKSIZE).

For variable-length spanned records processed under BSAM, specify LRECL=X, if logical record length exceeds 32,756.

For ASCII records on magnetic tape, the maximum record length is 2,048 bytes.

MODE Subparameter

The MODE subparameter specifies the mode of operation to be used with card reader, card punch, or card-read punch.

MODE= $\left\{ \begin{array}{l} C \left[\begin{array}{l} O \\ R \end{array} \right] \\ E \left[\begin{array}{l} O \\ R \end{array} \right] \end{array} \right\}$

C
indicates the card image (column binary) mode.

E
indicates the EBCDIC mode.

O
indicates optical mark read mode.

R
indicates read-column-eliminate mode.

Default: If you omit the MODE subparameter, E is assumed.

The subparameters MODE, CODE, KEYLEN, PRTSP, STACK, and TRTCH are mutually exclusive. Therefore, do not code any of these other subparameters with MODE.

Note: When the MODE subparameter is specified, either C or E must be specified if R is specified.

NCP Subparameter

The NCP subparameter specifies the maximum number of READ or WRITE macro instructions that can be issued before a CHECK macro instruction is issued to test for completion of the I/O operation.

NCP=number of channel programs

Default: If you omit the NCP subparameter, one is assumed.

The largest number that can be specified in the NCP subparameter is 99, but may be less depending on the amount of virtual storage available in the region or partition.

If chained scheduling is used, you must specify NCP as more than one.

OPTCD Subparameter

The OPTCD subparameter specifies the optional services that the control program performs.

OPTCD= $\left\{ \begin{array}{l} \{J\} \\ \{Z\} \\ \{B\} \\ \{T\} \\ \{U[C][J]\} \\ \{C[T][B][U]\} \\ \{H[Z][B]\} \\ \{W[C][T][B][U]\} \\ \{Z[C][T][B][U]\} \\ \{Q[C][T][B]\} \end{array} \right\}$

All optional services must be requested by one method. The characters may be coded in any order and when used in combination, no commas are permitted between characters.

These values can be used with the OPTCD subparameter:

B
requests that end-of-file recognition be treated as end-of-volume for tapes (permits SL and AL tapes to be read out of order).

C

requests that chained scheduling be used.

H

requests hopper empty exit for optical readers. Requests use of DOS/OS interchange feature for magnetic tape.

J (for 3800 only)

indicates that the first data byte is a table reference character that is used to select character arrangement tables.

Q

specifies that translation from ASCII input is required or that translation from EBCDIC to ASCII output is required.

T

requests user totaling facility.

U

For printers with the Universal Character Set feature and the 3800; unblocks data checks and allows analysis by an appropriate error analysis routine. If U is omitted, data checks are blocked, that is, not recognized as errors.

For Mass Storage System Extensions, Program Number 5740-XYG; specifies window processing to reduce the amount of staging space required to process large sequential data sets. These data sets must:

- reside on virtual DASD.
- have a physical sequential organization other than VSAM so that they can be processed by BSAM or QSAM.
- be allocated on a cylinder basis.
- be opened only for input or output.

For more information on window processing for sequential data sets, see *OS/VS Mass Storage System Extensions Services: Guide*, SH35-0035.

W

requests a validity check for write operations on direct access devices.

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 time. Use it only when a tape is known to be faulty and you don't have 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: specifies search direct (SD) for sequential data sets.

(text rearrangement only)

PRTSP Subparameter

The PRTSP subparameter specifies the line spacing on a printer as 0, 1, 2, or 3 lines between printout.

$$\text{PRTSP} = \left\{ \begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \end{array} \right\}$$

0
specifies that spacing is suppressed

1
specifies single spacing

2
specifies double spacing

3
specifies triple spacing.

Default: If you omit PRTSP information, 1 is assumed.

The PRTSP subparameter is valid only if the control characters A and M are not specified in the RECFM subparameter.

The subparameter 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 Subparameter

The RECFM subparameter specifies the format and characteristics of the records in the data set. The format and characteristics must be completely described by one source.

$$\text{RECFM} = \left\{ \begin{array}{l} \text{U} \left[\begin{array}{c} \text{T} \\ \text{A} \\ \text{M} \end{array} \right] \\ \text{V} \left[\begin{array}{c} \text{B} \\ \text{S} \\ \text{T} \\ \text{BS} \\ \text{BT} \end{array} \right] \left[\begin{array}{c} \text{A} \\ \text{M} \end{array} \right] \\ \text{F} \left[\begin{array}{c} \text{B} \\ \text{S} \\ \text{T} \\ \text{BS} \\ \text{BT} \end{array} \right] \left[\begin{array}{c} \text{A} \\ \text{M} \end{array} \right] \end{array} \right\}$$

For BSAM using ASCII data sets on tape:

$$\text{RECFM} = \left\{ \begin{array}{l} \text{D} \quad [\text{B}] \quad [\text{A}] \\ \text{U} \quad \quad \quad [\text{A}] \\ \text{F} \quad [\text{B}] \quad [\text{A}] \end{array} \right\}$$

A or M cannot be specified if the PRTSP subparameter is specified.

These values can be used with the RECFM subparameter:

A
indicates that the record contains ANSI control characters.

B
indicates that the records are blocked.

D
indicates that the data set contains variable length ASCII tape records.

F
indicates that the records are of fixed length.

M
indicates that the records contain machine code control characters.

S
for fixed-length records, the records are written as standard blocks, that is, 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.

T
indicates that the records may be written onto overflow tracks if required. Chained scheduling (OPTCD=C) cannot be used.

U
indicates that the records are of undefined length.

V
indicates that the records are of variable length.

Default: If you omit the RECFM subparameter, an undefined-length record is assumed with no optional features provided.

Notes:

- *RECFM=VBS does not provide the spanned record function; if this format is used, the problem program must block and segment the records.*
- *RECFM=VS or VBS cannot be specified for a SYSIN data set.*

STACK Subparameter

The STACK subparameter specifies which stacker bin receives a card.

$$\text{STACK} = \left\{ \begin{array}{l} 1 \\ 2 \end{array} \right\}$$

Default: If you omit the STACK information, a value of 1 is assumed.

The subparameters STACK, CODE, MODE, KEYLEN, PRTSP, and TRTCH are mutually exclusive subparameters. Therefore, if STACK is coded, do not code any of these other subparameters.

TRTCH Subparameter

The TRTCH subparameter specifies the recording technique for seven-track tape.

$$\text{TRTCH} = \left(\begin{array}{l} \text{C} \\ \text{E} \\ \text{T} \\ \text{ET} \end{array} \right)$$

C
specifies that the data conversion feature is to be used, with odd parity and no translation.

E
specifies even parity, with no translation and no conversion.

T
specifies that BCDIC to EBCDIC translation is required with odd parity and no data conversion feature.

ET

specifies even parity and no conversion, with BCDIC to EBCDIC translation required.

Default: If you omit TRTCH information, odd parity and no translation or data conversion are 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.

DCB Subparameters for BTAM

You can use the following DCB subparameters with BTAM:

BFTEK **EROPT**
BUFNO
DIAGNS
DSORG

The definitions and rules for coding these subparameters follow.

BFTEK Subparameter

The BFTEK subparameter specifies the type of buffering to be used by the control program.

BFTEK=D

D

indicates dynamic buffering in the processing program; if dynamic buffering is specified, a buffer pool must also be defined.

BUFNO Subparameter

The BUFNO subparameter specifies the number of buffers to be assigned to the DCB. The maximum is 255, but may be less than 255 because of limits established when the system was generated.

BUFNO=number of buffers

Requirements for coding the BUFNO subparameter are:

Method of obtaining the buffer pool:	Requirement for indicating the number of buffers:
BUILD macro instruction	BUFNO subparameter must be specified.
GETPOOL macro instruction	Control program uses the number specified in the GETPOOL macro instruction.
Automatically	Must be specified.

DIAGNS Subparameter

The DIAGNS subparameter specifies the OPEN/CLOSE/EOV trace option which gives a module-by-module trace of OPEN/CLOSE/EOV's work area and the DCB.

DIAGNS=TRACE

If the subparameter is not specified on the DD card, the option is not implemented. The GTF (Generalized Trace Facility) with the proper options specified, must be active in the system while the job that requested the trace is running. The options that must be specified for the GTF are MODE=EXT and TRACE=USR.

DSORG Subparameter

The DSORG subparameter specifies the organization of the data set and indicates whether the data set contains any location-dependent information that would make the data set unmovable.

The DSORG subparameter must always be coded on the DCB macro instruction.

DSORG=CX

CX

indicates a communications line group.

EROPT Subparameter

The EROPT subparameter requests the BTAM online terminal test option.

EROPT=T

T

requests the BTAM online terminal test option.

DCB Subparameters for EXCP

You can use the following DCB subparameters with EXCP:

BFALN	DEN	OPTCD
BFTEK	DIAGNS	PRTSP
BUFL	DSORG	STACK
BUFNO	KEYLEN	TRTCH
CODE	MODE	

The definitions and rules for coding these subparameters follow.

BFALN Subparameter

The BFALN subparameter specifies the boundary of each buffer.

$$\text{BFALN} = \left\{ \begin{array}{l} \text{F} \\ \text{D} \end{array} \right\}$$

F

indicates that each buffer starts on a fullword boundary that is not also doubleword boundary.

D

indicates that each buffer starts on a doubleword boundary.

Default: If BFALN is not specified, doubleword boundary alignment is assumed.

BFTEK Subparameter

The BFTEK subparameter specifies the type of buffering used by the control program.

$$\text{BFTEK} = \left\{ \begin{array}{l} \text{S} \\ \text{E} \end{array} \right\}$$

S

indicates simple buffering.

E

indicates exchange buffering.

BUFL Subparameter

The BUFL subparameter specifies the length, in bytes, of each buffer in the buffer pool. The maximum length that can be specified is 32,760 bytes.

BUFL=number of bytes

BUFNO Subparameter

The BUFNO subparameter specifies the number of buffers to be assigned to the DCB. The maximum is 255, but the actual number allowed may be less than 255 because of limits established when the system was generated.

BUFNO=number of buffers

Requirements for coding the BUFNO subparameter are:

Method of obtaining the buffer pool:

BUILD macro instruction
GETPOOL macro instruction

Requirement for indicating the number of buffers:

BUFNO subparameter must be specified.
Control program uses the number specified in the GETPOOL macro instruction.

CODE Subparameter

The CODE subparameter specifies the paper tape code in which the data is punched.

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.

$$\text{CODE} = \left\{ \begin{array}{c} \text{A} \\ \text{B} \\ \text{C} \\ \text{F} \\ \text{I} \\ \text{N} \\ \text{T} \end{array} \right\}$$

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).

Default: If the CODE subparameter is not specified, I is assumed.

DEN Subparameter

The DEN subparameter specifies the magnetic tape density in number of bpi (bits per inch) used to write a data set.

$$\text{DEN} = \left\{ \begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \right\}$$

0
for 7-track tape, indicates 200 bpi.

1
for 7-track tape, indicates 556 bpi.

2
for 7-track tape, and for 9-track tape, indicates 800 bpi.

3
for 9-track tape, indicates 1600 bpi.

4
for 9-track tape, indicates 6250 bpi.

Note: *When a tape is allocated to a 7-track drive with DEN=0 (200 bpi) for writing, the tape label and/or data may be written at 556 bpi instead. This occurs when a 3420 drive is allocated instead of a 2400 series or 3410. The 3420 executes the 200 bpi modeset as a 556 bpi modeset.*

Default: If the DEN subparameter is not specified:

800 bpi is assumed for 7-track tape and for 9-track NRZI tape without dual density.

1600 bpi is assumed for 9-track tape with 1600/800 bpi dual density or phase-encoded tape.

6250 bpi is assumed for 9-track tape with 6250/1600 bpi dual density or group coded recording tape.

For 7-track tape, all information on the reel must be written in the same density; that is, labels, data, tapemarks.

If not specified by any source, the highest applicable density is assumed.

DIAGNS Subparameter

The DIAGNS subparameter specifies the OPEN/CLOSE/EOV trace option which gives a module-by-module trace of OPEN/CLOSE/EOV's work area and the DCB.

DIAGNS=TRACE

If the subparameter is not specified on the DD card, the option is not implemented. The GTF (Generalized Trace Facility), with the proper options specified, must be active in the system while the job that requested the trace is running. The options that must be specified for the GTF are MODE=EXT and TRACE=USR.

DSORG Subparameter

The DSORG subparameter specifies the organization of the data set and indicates whether the data set contains any location-dependent information that would make the data set unmovable. The DSORG subparameter must always be coded on the DCB macro instruction.

DSORG=data set organization

You can code these values with the DSORG subparameter:

DA
indicates direct organization data set.

IS
indicates indexed sequential data set.

PO
indicates partitioned organization data set.

PS
indicates physical sequential data set.

KEYLEN Subparameter

The KEYLEN subparameter specifies the length, in bytes, of the keys used in the data set.

KEYLEN=number of bytes

The subparameters KEYLEN, CODE, DEN, MODE, PRTSP, STACK, and TRTCH are mutually exclusive subparameters. Therefore, if you code KEYLEN, do not code any of these other subparameters.

MODE Subparameter

The MODE subparameter specifies the mode of operation used with a card reader, a card punch, or a card-read punch.

MODE= { C }
 { E }

C
indicates the card image (column binary) mode.

E
indicates the EBCDIC mode.

Default: If you code the MODE subparameter, E is assumed. The subparameters MODE, CODE, KEYLEN, PRTSP, and TRTCH are mutually exclusive. Therefore, do not code any of these other subparameters with MODE.

OPTCD Subparameter

The OPTCD subparameter specifies the optional services performed by the control program.

OPTCD=Z

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 time. It should be used only when a tape is known to be faulty and when 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: specifies search direct (SD) for sequential data sets.

PRTSP Subparameter

The PRTSP subparameter specifies the line spacing on a printer as 0, 1, 2, or 3 lines between printout.

$$\text{PRTSP} = \left\{ \begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \end{array} \right\}$$

0
specifies that spacing is suppressed

1
specifies single spacing

2
specifies double spacing

3
specifies triple spacing.

Default: If you omit PRTSP information, 1 is assumed.

The PRTSP subparameter is valid only if the control characters A and M are not specified in the RECFM subparameter.

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.

STACK Subparameter

The STACK subparameter specifies which stacker bin receives a card.

$$\text{STACK} = \left\{ \begin{array}{c} 1 \\ 2 \end{array} \right\}$$

Default: If you omit STACK information, a value of 1 is assumed.

The subparameters STACK, CODE, DEN, KEYLEN, MODE, PRTSP, and TRTCH are mutually exclusive subparameters. Therefore, if you code STACK, do not code any of these other subparameters.

TRTCH Subparameter

The TRTCH subparameter specifies the recording technique for 7-track tape.

$$\text{TRTCH} = \left\{ \begin{array}{c} C \\ E \\ T \\ ET \end{array} \right\}$$

C
specifies that the data conversion feature is used, with odd parity and no translation.

E

specifies even parity, with no translation and conversion.

T

specifies odd parity and no conversion, and that BCD to EBCDIC translation is required when reading, and EBCDIC to BCD translation is required when writing.

ET

specifies even parity and no conversion, and that BCD to EBCDIC translation is required when reading, and EBCDIC to BCD translation is required when writing.

Default: If you omit TRTCH information, odd parity and no translation or data conversion are 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.

DCB Subparameters for GAM

You can use these DCB subparameters with GAM:

DIAGNS
DSORG
GNCP

The definitions and rules for coding these subparameters follow.

DIAGNS Subparameter

The DIAGNS subparameter specifies the OPEN/CLOSE/EOV trace option which gives a module-by-module trace of OPEN/CLOSE/EOV's work area and the DCB.

DIAGNS=TRACE

If the subparameter is not specified on the DD card, the option is not implemented. The GTF (Generalized Trace Facility), with the proper options specified, must be active in the system while the job that requested the trace is running. The options that must be specified for the GTF are MODE=EXT and TRACE=USR.

DSORG Subparameter

The DSORG subparameter specifies the organization of the data set and indicates whether the data set contains any location-dependent information that would make the data set unmovable.

The DSORG subparameter must always be coded on the DCB macro instruction.

DSORG=GS

GS
graphic data control block.

GNCP Subparameter

The GNCP subparameter specifies the maximum number of input/output macro instructions that will be issued before a WAIT macro instruction.

GNCP=number of macros

Default: If you omit the GNCP subparameter, a value of one is assumed.

The value of GNCP must be from 1 to 99 at execution time.

The subparameters GNCP, BFTEK, and BFALN are mutually exclusive subparameters. Therefore, do not code any of these other subparameters with GNCP.

For further information on the GNCP subparameter, refer to *OS/VS Graphic Programming Services (GPS) for IBM 2250 Display Unit*, GC27-6971.

DCB Subparameter for QISAM

You can use these DCB subparameters with QISAM:

BFALN	DIAGNS	OPTCD
BLKSIZE	DSORG	RECFM
BUFL	KEYLEN	RKP
BUFNO	LRECL	
CYLOFL	NTM	

The definitions and rules for coding these subparameters follow.

BFALN Subparameter

The BFALN subparameter specifies the boundary of each buffer.

BFALN= { F }
 { D }

F

indicates that each buffer starts on a fullword boundary that is not also a doubleword boundary.

D

indicates that each buffer starts on a doubleword boundary.

Default: If you omit BFALN, doubleword boundary alignment is assumed.

BLKSIZE Subparameter

The BLKSIZE subparameter specifies the maximum length, in bytes, of a block. The maximum length that can be specified in the BLKSIZE subparameter is 32,760. BLKSIZE must be specified when creating indexed sequential data sets. BLKSIZE must be omitted when processing existing indexed sequential data sets.

BLKSIZE=number of bytes

Blocksize varies according to record format (specified by RECFM subparameter).

- If RECFM=F, BLKSIZE must be logical record length.
- If RECFM=FB, BLKSIZE must be an integral multiple of the logical record length.
- If RECFM=V, BLKSIZE must be (maximum logical record length + 4).
- If RECFM=VB, BLKSIZE must be (maximum block size + 4).

The blocksize 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. Blocksize information is required only when creating a data set containing block records.

BUFL Subparameter

The BUFL subparameter specifies the length, in bytes, of each buffer in the buffer pool. The maximum number that can be specified is 32,760.

BUFL=number of bytes

Default: The system acquires buffers automatically.

BUFNO Subparameter

The BUFNO subparameter specifies the number of buffers assigned to the DCB. The maximum is 255, but the actual number allowed may be less than 255 because of limits established when the system was generated.

BUFNO=number of buffers

Requirements for coding BUFNO subparameter are:

Method of obtaining the buffer pool:	Requirement for indicating the number of buffers:
BUILD macro instruction	BUFNO subparameter must be specified.
GETPOOL macro instruction	Control program uses the number specified in the GETPOOL macro instruction.
Automatically	Optional; if not specified, two buffers are obtained.

CYLOFL Subparameter

The CYLOFL subparameter specifies how many tracks on each cylinder are to hold the records that overflow from other tracks on that cylinder. The maximum number that can be specified is 99. The CYLOFL subparameter is to be used only when OPTCD=Y.

CYLOFL=number of tracks

DIAGNS Subparameter

The DIAGNS subparameter specifies the OPEN/CLOSE/EOV trace option, which gives a module-by-module trace of OPEN/CLOSE/EOV's work area and the DCB.

DIAGNS=TRACE

If the subparameter is not specified on the DD card, the option is not implemented. The GTF (Generalized Trace Facility), with the proper options specified, must be active in the system while the job that requested the trace is running. The options that must be specified for the GTF are MODE=EXT and TRACE=USR.

DSORG Subparameter

The DSORG subparameter specifies the organization of the data set and indicates whether the data set contains any location-dependent information that would make the data set unmovable. The DSORG subparameter must be coded on the DD statement that defines the data set.

DSORG=data set organization

You can code the following data set organizations with the DSORG subparameter:

IS
indexed sequential.

ISU

indexed sequential location dependent information.

ISU can be specified only when an ISAM data set is being created.

KEYLEN Subparameter

The KEYLEN subparameter specifies the length, in bytes, of the keys used in an indexed sequential data set.

KEYLEN=number of bytes

KEYLEN can only be specified when indexed sequential data sets are created. KEYLEN must be omitted when processing existing indexed sequential data sets.

LRECL Subparameter

The LRECL subparameter specifies the length, in bytes, for fixed-length records or it specifies the maximum length, in bytes, for variable length records.

LRECL=number of bytes

The length cannot exceed the blocksize (BLKSIZE).

- If RECFM=V or VB, LRECL must be equal to the largest logical record length + 4.
- If RECFM=F or FB, LRECL must be equal to the logical record length.

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 the data set is being created.

NTM Subparameter

The NTM subparameter specifies the number of tracks used for a cylinder index. When the specified number of tracks is filled, a master index is created. This information is required only when the master index option (OPTCD=M) has been selected.

NTM=number of tracks

If you omit NTM information and OPTCD=M is specified, the master index option is ignored.

OPTCD Subparameter

The OPTCD subparameter specifies the optional services performed by the control program.

OPTCD={ [I] [L] [M] [R] [U] [W] [Y] }

All optional services must be requested by one method. The characters may be coded in any order and no commas are permitted between characters.

These values can be used with the OPTCD subparameter:

I

requests that the control program use the independent overflow areas for overflow records.

L Informs the control program that the user may indicate deleted records by placing all one's in the first byte of fixed length records, or the fifth byte of variable length records. These records are physically deleted when bumped to the overflow area. This option cannot be specified for blocked fixed-length records if RKP=0 or for variable-length records if RKP=4.

M requests that the system create and maintain a master index(es) according to the number of tracks specified in the NTM subparameter.

R requests the control program to place reorganization criteria information in certain fields of the DCB. The program can analyze these statistics to determine when to reorganize the data set.

This option is provided whenever the OPTCD subparameter is omitted from all sources.

U specifies that the system accumulates track index entries in storage and writes them as a group for each track of the track index. OPTCD=U can only be specified for fixed-length records.

W requests a validity check for write operations on direct access devices.

Y requests that the system use the cylinder overflow areas for overflow records.

RECFM Subparameter

The RECFM subparameter specifies the format and characteristics of the records in the data set. The format and characteristics must be completely described by one source.

$$\text{RECFM} = \left\{ \begin{array}{l} \text{V[B]} \\ \text{F[B]} \end{array} \right\}$$

These values can be used with the RECFM subparameter:

B indicates that the records are blocked.

F indicates that the records are of fixed length.

V indicates that the records are of variable length; variable-length records cannot be in ASCII.

Default: If you omit the RECFM subparameter, a variable-length record is assumed.

When indexed sequential data sets are created, you can code the RECFM subparameter; when existing indexed sequential data sets are processed, RECFM must be omitted.

RKP Subparameter

The RKP subparameter 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.

RKP=number

Default: If you omit RKP information, a relative key position of zero is assumed.

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 four or greater, when the delete option (OPTCD=L) is not specified. The relative key position must be five or greater if the delete option is specified.

DCB Subparameters for QSAM

You can use these DCB subparameters with QSAM:

BFALN	BUFOFF	EROPT	PRTSP
BFTEK	CODE	FUNC	RECFM
BLKSIZE	DEN	LRECL	STACK
BUFL	DIAGNS	MODE	TRTCH
BUFNO	DSORG	OPTCD	

The definitions and rules for coding these subparameters follow.

BFALN Subparameter

The BFALN subparameter specifies the boundary of each buffer.

$$\text{BFALN} = \left\{ \begin{array}{l} \text{F} \\ \text{D} \end{array} \right\}$$

F

indicates that each buffer starts on a fullword boundary that is not also a doubleword boundary.

D

indicates that each buffer starts on a doubleword boundary.

Default: If BFALN is not specified by any source, doubleword boundary alignment is assumed.

The buffer alignment information (BFALN subparameter) must be supplied by the same source as the type of buffering (BFTEK subparameter) or both subparameters must be omitted.

BFTEK Subparameter

The BFTEK subparameter specifies the type of buffering used by the control program.

$$\text{BFTEK} = \left\{ \begin{array}{l} \text{S} \\ \text{E} \\ \text{A} \end{array} \right\}$$

S

indicates simple buffering.

E

indicates exchange buffering.

Exchange buffering cannot be specified for variable-length blocked records or spanned records.

A

indicates a logical record interface for variable-length spanned records.

Default: If BFTEK information is not specified by any source, simple buffering (S) is assumed.

BLKSIZE Subparameter

The BLKSIZE subparameter specifies the maximum length, in bytes, of a block. For magnetic tape the minimum number is 18. The largest number that can be specified in the BLKSIZE subparameter is 32,760. However, for blocks of ASCII records on magnetic tape, the maximum number is 2,048.

BLKSIZE=number of bytes

- If RECFM=F, BLKSIZE must be logical record length.
- If RECFM=FB, BLKSIZE must be an integral multiple of the logical record length.
- If RECFM=V, BLKSIZE must be (maximum record size + 4).
- If RECFM=VB, BLKSIZE must be (at least four greater than maximum logical record length).
- If RECFM=D or RECFM=DB, BLKSIZE must be (at least maximum record length + block prefix length).

If you code the BLKSIZE subparameter in the DCB macro instruction or on a DD statement that defines an existing data set with standard labels, the subparameter overrides the blocksize specified in the label.

If you are using a 3886 optical card reader, you must specify a block size of 16 times LRECL.

BUFL Subparameter

The BUFL subparameter specifies the length, in bytes, of each buffer in the buffer pool. The maximum length that can be specified is 32,760 bytes.

BUFL=number of bytes

The BUFL subparameter is optional; if you omit it and the control program acquires buffers automatically, the blocksize and key length information is used to establish buffer length. If card image is specified (MODE=C), BUFL=160 must be specified.

BUFNO Subparameter

The BUFNO subparameter specifies the number of buffers assigned to the DCB. The maximum number is 255, but may be less than 255 because of the size of the partition.

BUFNO=number of buffers

Requirements for coding the BUFNO subparameter are:

Method of obtaining the buffer pool:	Requirement for indicating the number of buffers:
BUILD macro instruction	BUFNO subparameter must be specified. Control program uses the number specified in the GETPOOL macro instruction. Optional; if not specified, two buffers are obtained for a non-unit record device, and three for a unit record device.
GETPOOL macro instruction	
Automatically	

BUFOFF Subparameter

The BUFOFF subparameter specifies the buffer offset. The buffer offset is the length of an optional block prefix that can precede a block of one or more ASCII records on magnetic tape.

$$\text{BUFOFF} = \left\{ \begin{array}{c} n \\ L \end{array} \right\}$$

n
is a number that indicates 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
indicates that the block prefix field is four bytes long and contains the block length. **L** may be specified only when record format (RECFM) is D or DB.

CODE Subparameter

The CODE subparameter specifies the paper tape code in which the data is punched.

The subparameters CODE, KEYLEN, MODE, PRTSP, STACK, and TRTCH are mutually exclusive subparameters. Therefore, if you use CODE, do not use any of these other subparameters.

$$\text{CODE} = \left\{ \begin{array}{c} A \\ B \\ C \\ F \\ I \\ N \\ T \end{array} \right\}$$

A
ASCII (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).

Default: If you omit the CODE subparameter, I is assumed.

DEN Subparameter

The DEN subparameter specifies the magnetic tape density in number of bpi (bits per inch) used to write a data set.

$$\text{DEN} = \left\{ \begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \right\}$$

0
for 7-track tape, indicates 200 bpi.

1
for 7-track tape, indicates 556 bpi.

2
for 7-track tape, and for 9-track tape, indicates 800 bpi.

3
for 9-track tape, indicates 1600 bpi.

4
for 9-track tape, indicates 6250 bpi.

Note: *When a tape is allocated to a 7-track drive with DEN=0 (200 bpi) for writing, the tape label and/or data may be written at 556 bpi instead. This occurs when a 3420 drive is allocated instead of a 2400 series or 3410. The 3420 executes the 200 bpi modeset as a 556 bpi modeset.*

Default: If the DEN subparameter is not specified:

800 bpi is assumed for 7-track tape and for 9-track NRZI tape without dual density.

1600 bpi is assumed for 9-track tape with 1600/800 bpi dual density or phase-encoded tape.

6250 bpi is assumed for 9-track tape with 6250/1600 bpi dual density or group coded recording tape.

For 7-track tape, all information on the reel must be written in the same density; that is, labels, data, tapemarks.

If not specified by any source, the highest applicable density is assumed.

The subparameters CODE, DEN, KEYLEN, MODE, PRTSP, and STACK are mutually exclusive subparameters. Therefore, if you use DEN, do not use any of these other subparameters.

DIAGNS Subparameter

The DIAGNS subparameter specifies the OPEN/CLOSE/EOV trace option which gives a module-by-module trace of OPEN/CLOSE/EOV's work area and the DCB.

DIAGNS=TRACE

If the subparameter is not specified on the DD card, the option is not implemented. The GTF (Generalized Trace Facility), with the proper options specified, must be active in the system while the job that requested

the trace is running. The options that should be specified for the GTF are MODE=EXT and TRACE=USR.

DSORG Subparameter

The DSORG subparameter specifies the organization of the data set and indicates whether the data set contains any location-dependent information that would make the data set unmovable.

The DSORG subparameter must always be coded on the DCB macro instruction.

DSORG=data set organization

You can code the following data set organizations with the DSORG subparameter:

PS

physical sequential data set.

PSU

physical sequential data set that contains location dependent information.

EROPT Subparameter

The EROPT subparameter specifies the option executed if an error occurs in reading or writing a record.

EROPT= { ACC
 SKP
 ABE }

ACC

indicates that the system accepts the block causing the error.

SKP

indicates that the system skips the block causing the error.

ABE

indicates that the system causes an ABEND.

Default: If you omit EROPT subparameter, ABE is assumed.

FUNC Subparameter

The FUNC subparameter specifies the type of data set opened for the 3525 Card Punch.

FUNC=function

You can code these functions with the FUNC subparameter:

I

data in a data set is to be punched into and printed on cards.

R

data set is for reading cards.

P

data set is for punching cards.

W
data set is for printing

D
data protection for a punch data set

X
printer

T
two-line printer.

The only valid combinations of these values are:

I	WT	RWT	RPW
R	RP	PW	RPWXT
P	RPD	PWXT	RPWD
W	RW		

Default: If the function information is not supplied by any source, P is assumed for output results and R is assumed for input.

LRECL Subparameter

The LRECL subparameter specifies the actual or maximum length, in bytes, of a logical record.

LRECL= { n }
 { X }

n
number of bytes

X
specified for variable-length spanned records that exceed 32,760.

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 blocksize (BLKSIZE) except for variable-length spanned records.

- If RECFM=V or VB, LRECL must be equal to the largest logical record length + 4.
- If RECFM=F or FB, LRECL is the logical record length.
- If RECFM=U, LRECL should be omitted.
- If RECFM=D or DB, LRECL must be equal to the maximum record length.

For variable-length spanned records (VS or VBS) processed under QSAM in GET or PUT locate mode, if the logical record size exceeds 32,760, LRECL=X must be specified.

For ASCII records on magnetic tape, the maximum record length is 2,048 bytes.

MODE Subparameter

The MODE subparameter specifies the mode of operation used with a card reader, a card punch, or a card-read punch.

$$\text{MODE} = \left\{ \begin{array}{l} \text{C} \quad [\text{O}] \\ \text{E} \quad [\text{R}] \end{array} \right\}$$

C
indicates the card image (column binary) mode.

E
indicates the EBCDIC mode.

O
indicates optical mark read mode.

R
indicates read column eliminate mode.

Default: If you omit the MODE subparameter, E is assumed.

The subparameters MODE, CODE, DEN, KEYLEN, PRTSP, STACK, and TRTCH are mutually exclusive. Therefore, do not code any of these other subparameters with MODE.

OPTCD Subparameter

The OPTCD subparameter specifies the optional services to be performed by the control program.

$$\text{OPTCD} = \left\{ \begin{array}{l} \{\text{J}\} \\ \{\text{Z}\} \\ \{\text{B}\} \\ \{\text{T}\} \\ \{\text{U}[\text{C}][\text{J}]\} \\ \{\text{C}[\text{T}][\text{B}][\text{U}]\} \\ \{\text{H}[\text{Z}][\text{B}]\} \\ \{\text{W}[\text{C}][\text{T}][\text{B}][\text{U}]\} \\ \{\text{Z}[\text{C}][\text{T}][\text{B}][\text{U}]\} \\ \{\text{Q}[\text{C}][\text{T}][\text{B}]\} \end{array} \right\}$$

All optional services must be requested by one method. The characters may be coded in any order and when used in combination, no commas are permitted between characters.

These values can be used with the OPTCD subparameter:

B
requests that end-of-file recognition be treated as end-of volume for tapes (permits SL and AL tapes to be read out of order).

C
requests that chained scheduling be used.

H
requests hopper empty exit for optical readers. Requests use of DOS/OS interchange feature for magnetic tape.

J (for the 3800 only)
indicates that the first byte of each output data line is a table reference character that is used to select a character arrangement table.

Q
specifies that translation from ASCII input is required or that translation from EBCDIC to ASCII output is required.

T
requests user totaling facility. T cannot be specified for a SYSIN or SYSOUT data set.

U
for printers with the Universal Character Set feature and the 3800; unblocks data checks and allows analysis by an appropriate error analysis routine. If U is omitted, data checks are blocked, that is, not recognized as errors.

For Mass Storage System Extensions, Program Number 5740-XYG; specifies window processing to reduce the amount of staging space required to process large sequential data sets. These data sets must:

- reside on virtual DASD.
- have a physical sequential organization other than VSAM so that they can be processed by BSAM or QSAM.
- be allocated on a cylinder basis.
- be opened only for input or output.

For more information on window processing for sequential data sets, see *OS/VS Mass Storage System Extensions Services: Guide*, SH35-0035.

W
requests a validity check for write operations on direct access devices.

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 time. 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: specifies search direct (SD) for sequential data sets.

PRTSP Subparameter

The PRTSP subparameter specifies the line spacing on a printer as 0, 1, 2, or 3 lines between printout.

$$\text{PRTSP} = \left\{ \begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \end{array} \right\}$$

0
specifies that spacing is suppressed

1
specifies single spacing

2
specifies double spacing

3
specifies triple spacing.

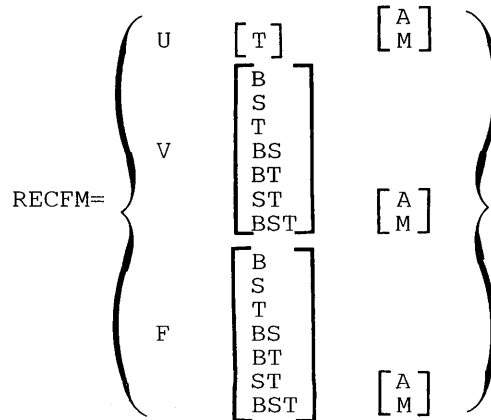
Default: If you omit PRTSP information, 1 is assumed.

The PRTSP subparameter is valid only if the control characters A and M are not specified in the RECFM subparameter.

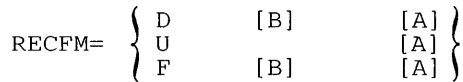
The subparameters PRTSP, CODE, KEYLEN, MODE, STACK, and TRTCH are mutually exclusive subparameters. Therefore, if you code PRTSP, do not code any of these other subparameters.

RECFM Subparameter

The RECFM subparameter specifies the format and characteristics of the records in the data set. The format and characteristics must be completely described by one source.



With ASCII data sets on tape:



A or M cannot be specified if the PRTSP subparameter is specified.

These values can be used with the RECFM subparameter:

A

indicates that the record contains ANSI printer control characters.

B

indicates that the records are blocked.

D

specifies that the data set contains variable length ASCII tape records, and they are written according to American National Standards (ASCII).

F

indicates that the records are of fixed length.

M

indicates that the records contain machine code control characters.

S

for fixed-length records, the records are written as standard blocks, that is, 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 cannot be specified.

T

indicates that the records may be written onto overflow tracks if required. Exchange buffering or chained scheduling (OPTCD=C) cannot be used.

U

indicates that the records are of undefined length.

V

indicates that the records are of variable length.

Default: If you omit RECFM subparameter, an undefined-length record is assumed with no optional features provided.

Note: RECFM=VS or VBS cannot be specified for a SYSIN data set.

STACK Subparameter

The STACK subparameter specifies which stacker bin receives a card.

$$\text{STACK} = \left\{ \begin{array}{c} 1 \\ 2 \end{array} \right\}$$

Default: If you omit STACK information, a value of 1 is assumed.

The subparameter STACK, CODE, DEN, KEYLEN, MODE, PRTSP, and TRTCH are mutually exclusive subparameters. Therefore, if you code STACK, do not code any of these other subparameters.

TRTCH Subparameter

The TRTCH subparameter specifies the recording technique for 7-track tape.

$$\text{TRTCH} = \left(\begin{array}{c} C \\ E \\ T \\ ET \end{array} \right)$$

C

specifies that the data conversion feature is used, with odd parity and no translation.

E

specifies even parity, with no translation and no conversion.

T

specifies that BCDIC to EBCDIC translation is required with odd parity and no data conversion feature.

ET

specifies that even parity and no conversion, with BCDIC to EBCDIC translation required.

Default: If you omit TRTCH information, odd parity and no translation or data conversion are assumed.

The subparameters TRTCH, CODE, KEYLEN, MODE, PRTSP, and STACK are mutually exclusive subparameters. Therefore, if you code TRTCH, do not code any of these other subparameters.

DCB Subparameters for TCAM

These DCB subparameters can be used with TCAM:

BLKSIZE	BUFSIZE	RECFM
BUFIN	INTVL	RESERVE
BUFL	LRECL	THRESH
BUFMAX	OPTCD	
BUFOUT	PCI	

The definitions and rules for coding these subparameters follow.

BLKSIZE Subparameter

The BLKSIZE subparameter specifies the length in bytes of the application program's work area into which TCAM will move message units to be processed.

BLKSIZE=number of bytes

The number specified should be at least equal to the record length as specified by the LRECL operand; the maximum number that can be specified is 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 identify the message segment.

For variable-length records, four bytes must be included for unblocked records, or eight bytes for blocked records.

BUFIN Subparameter

The BUFIN subparameter specifies the number of buffers to be assigned initially for receiving operations for each line in the line group.

BUFIN=number of buffers

Default: If you omit the BUFIN subparameter, one is assumed.

The number specified in the BUFIN subparameter must be less than the number of buffers in the buffer pool for this line group; the number cannot exceed 15.

The sum of buffers specified in the BUFIN and BUFOUT subparameters must be no greater than the number of buffers in the buffer pool for this line group, not including those for disk activity only.

BUFL Subparameter

The BUFL subparameter specifies the length, in bytes, of each of the MCP (message control program) buffers that handle messages received and sent by an application program.

BUFL=number of bytes

The number of bytes must be at least 31, but cannot exceed 65,535.

BUFMAX Subparameter

The BUFMAX subparameter specifies the maximum number of buffers allocated to a line at one time. The number specified must be greater than one but cannot exceed 15. The number must be at least equal to the larger of the numbers specified by the BUFIN and BUFOUT subparameters.

BUFMAX=number of buffers

Default: If you omit the BUFMAX subparameter, two is assumed.

BUFOUT Subparameter

The BUFOUT subparameter specifies the number of buffers initially assigned for sending operations for each line in the line group.

BUFOUT=number of buffers

Default: If you omit the BUFOUT subparameter, two is assumed. The number specified must be less than the number of buffers in the buffer pool for this line group and cannot 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).

BUFSIZE Subparameter

The BUFSIZE subparameter specifies the length, in bytes, of each of the buffers used for all lines in a particular line group. This length must be at least 31 bytes, but cannot exceed 65,535.

The buffer size should be an even multiple of the buffer unit size as specified in the INTRO macro instruction; the maximum number of buffer units per buffer is 255.

BUFSIZE=number of bytes

INTVL Subparameter

Specifies the polling interval (that is, the number of seconds of intentional delay between passes through a polling list) for the lines in this line group.

INTVL=number of seconds

Default: If you omit the INTVL subparameter, zero is assumed.

After all the terminals in a polling list for a give 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 of seconds specified must not exceed 255.

This subparameter must be omitted if the line group consists of switched lines.

LRECL Subparameter

The LRECL subparameter specifies the actual or maximum length, in bytes, of a logical record.

LRECL=number of bytes

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 blocksize (BLKSIZE) except for variable-length spanned records.

- If RECFM=V or VB, LRECL must be equal to the largest logical record length + 4.
- If RECFM=F or FB, LRECL must be equal to the largest logical record length.
- If RECFM=U, LRECL should be omitted.

The record length should include the source and control bytes if these are specified in the OPTCD subparameter. The record length is required for fixed-length records only.

OPTCD Subparameter

The OPTCD subparameter specifies the optional services performed by the control program.

$$\text{OPTCD} = \left\{ \begin{array}{c} \text{C} \\ \text{U} \\ \text{W} \end{array} \right\}$$

All optional services must be requested by the same source. The characters may be coded in any order and no commas are permitted between characters.

These values can be used with the OPTCD subparameter:

C
specifies that one byte of the work area be used to indicate if a segment of a message is the first, middle, or last segment.

U
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
specifies that the name of each message source be placed in an eight-byte field in the work area.

PCI Subparameter

The PCI (program check interruption) subparameter specifies whether a PCI is to be used to control the allocation and freeing of buffers; the PCI subparameter also specifies how these operations are to be performed.

$$\text{PCI} = \left(\begin{array}{c} \text{N} \\ \text{R} \\ \text{A} \\ \text{X} \end{array} \right) \left(\begin{array}{c} \text{,N} \\ \text{,R} \\ \text{,A} \\ \text{,X} \end{array} \right)$$

The operands shown in the preceding format 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.

R

specifies that after the first buffer is filled (on receiving operations) or emptied (on sending operations), a PCI occurs during the filling and 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, and a buffer is allocated in its place.

X

specifies that after a buffer is filled (on receive operations) or emptied (on send operations), a PCI occurs during filling or emptying of the next buffer. The first buffer is not deallocated, but a new buffer is allocated. Buffer deallocation occurs at the end of transmission, or when EOB/ETB control character is sent, if EOB/ETB checking is specified in STARTMH macro.

PCI=X (or N) must be used if TCAM network defines logical messages, and if SETOM specified PROCESS=YES, to insure that logical messages are not deblocked until block checking is performed; otherwise, a logical message containing an error could be routed to its destination.

Default: If you omit the PCI subparameter, PCI=(A,A) is assumed.

RECFM Subparameter

The RECFM subparameter specifies the format and characteristics of the records in the data set. The format and characteristics must be completely described by one source.

$$\text{RECFM} = \left\{ \begin{array}{l} \text{U} \\ \text{V} \\ \text{F} \end{array} \right\} \quad [\text{B}]$$

These values can be used with the RECFM subparameter:

B

indicates that the records are blocked.

F

indicates that the records are of fixed length.

U

indicates that the records are of undefined length.

V

indicates that the records are of variable length.

RESERVE Subparameter

The RESERVE subparameter 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.

$$\text{RESERVE} = (\text{number 1}, \text{number 2})$$

number 1

indicates the number of bytes reserved in the first buffer that receives an incoming message.

number 2

indicates the number of bytes reserved in all the buffers following the first buffer in a multiple-buffer header situation.

THRESH Subparameter

The THRESH subparameter specifies the percentage of the nonreusable disk message queue records to be used before a flush closedown occurs.

THRESH=number

If you omit the THRESH subparameter, closedown occurs when 95% of the records have been used.

DDNAME Parameter

Keyword,Optional

The DDNAME parameter allows you to postpone defining a data set until later in the same job step. In the case of cataloged procedures, this parameter allows you to postpone defining a data set in the procedure until the procedure is called by a job step.

For additional information on the ddname parameter, see Appendix A of this publication.

DDNAME=ddname

ddname

the name of the DD statement on which the data set will be defined.

Rules for Coding

1. The only parameters you can code 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. 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.

Examples of the DDNAME Parameter

```
//STEP1 EXEC PGM=PROGRAM8
//DD1 DD DDNAME=INPUT
//DD2 DD DSN=NAME=WELL,DISP=OLD
```

The preceding statements comprise 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
.
.
.
/*
```

```
//ST4 EXEC PGM=FIFTY
//DD1 DD DDNAME=DD5
//DD2 DD UNIT=2400
//DD3 DD UNIT=2400
//DD4 DD SYSOUT=B
//DD5 DD DSN=ADDN,DISP=(,PASS),UNIT=2400
//ST5 EXEC PGM=FINE
//DD6 DD DSN=*.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 requests that the system 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.

```
//STEP8 EXEC PGM=BLOCK
//DD1 DD DDNAME=SKIP
// DD DSN=A.B.C,DISP=OLD
// DD DSN=LEV.FIVE,DISP=OLD
//SKIP DD DSN=SAK,DISP=OLD,UNIT=2314,
// 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 SAK.

```
//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.

DEST Parameter (for RES)

Keyword, Optional

RES (remote entry services) provides the facility to submit jobs to a central computing center from a remote workstation and to route output to remote workstations.

The DEST parameter specifies a remote destination (workstation) for an output data set.

For further information on RES, see *OS/VSI RES Workstation User's Guide*, GC28-6879.

DEST=userid

userid

indicates a remote destination for an output data set.

Rules for Coding

1. Code a valid userid established by your installation. The userid must consist of one to seven alphanumeric characters.
2. The DEST parameter must be coded with the SYSOUT parameter on the DD statement.
3. *Default:* If you do not code the DEST parameter, the default destination is the workstation from which the job was submitted.

If the userid you specified is invalid, or if the output cannot be routed to that destination due to protection masks, the default destination is assumed and an appropriate message is issued.

Example of the DEST Parameter

```
//JOB01      JOB          , 'REBECCA BARNHARDT' ,MSGLEVEL=1
//STEP01     EXEC        PGM=INTEREST
//DEB        DD          SYSOUT=A
//GWB        DD          SYSOUT=A,DEST=STAT04
```

In this example, the workstation from which the job was submitted receives the output described by the DEB DD statement. The user identified by the userid STAT04 receives the output described by the GWB DD statement.

DISP Parameter

Keyword,Optional

The DISP parameter describes the status of a data set to the system. It also indicates what is to be done with the data set after termination of the job step or job that processes it. 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 ABENDS (conditional disposition). Figure 10 contains information about disposition processing.

For further information on the DISP parameter including data set integrity, see *OS/VS1 JCL Services*, GC24-5100.

```
DISP=( [ NEW ] [ ,DELETE ] [ ,DELETE ] )  
       [ OLD ] [ ,KEEP ] [ ,KEEP ]  
       [ SHR ] [ ,PASS ] [ ,CATLG ]  
       [ SHARE ] [ ,CATLG ] [ ,UNCATLG ]  
       [ MOD ] [ ,UNCATLG ]
```

NEW

specifies that the data set is created in this job step.

OLD

specifies that the data set existed before this job step.

SHR or SHARE

specifies that the data set existed before this job step and can be used simultaneously (shared) by another job.

MOD

specifies that the read/write mechanism is 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 kept on the volume at the end of this job step.

,PASS

specifies that the data set is passed for use by a subsequent job step in the same job.

,CATLG

specifies that the data set is kept at the end of this job step and an entry pointing to the data set is placed in the system catalog. Any missing index levels will be created.

,UNCATLG

specifies that the data set is kept at the end of this job step, but the entry pointing to the data set in the system catalog and unneeded indexes, with the exception of the highest level index, are removed.

specifies no explicit disposition for the data set, but indicates that 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 released for use by other data sets if this step ABENDS.

,KEEP

specifies that the data set is kept on the volume if this step ABENDS.

,CATLG

specifies that an entry pointing to the data set is placed in the system catalog if this step ABENDS. Any missing index levels will be created.

,UNCATLG

specifies that the entry pointing to the data set in the system catalog and unneeded indexes, with the exception of the highest level index, are removed if this step ABENDS.

Rules for Coding

1. If you code only the first subparameter, you may omit the 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, omit 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. If MSVGP is coded with DISP=OLD or DISP=SHR, the MSVGP information will be ignored.
6. You must specify a disposition of PASS or DELETE for a data set with a system-generated name; that is, when DSNAME=dsname is omitted from the DD statement. Any other disposition is overridden by the system with PASS.

Examples of the DISP Parameter

```
//DD      DD      DSNAME=D99.GROUP.SIX,UNIT=2314,  
//          VOLUME=SER=111111,  
//          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. Because the data set's name is qualified, the system automatically creates all necessary index levels.

```
//DD2     DD      DSNAME=FIX,UNIT=2400-1,  
//          VOLUME=SER=44889,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 that the system delete the data set if the step ABENDS.

```
//STEP1   EXEC    PGM=FULL  
//DD1     DD      DSNAME=SWITCH.LEVEL18.GROUP12,  
//          UNIT=2314,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 ABENDS, the data set is deleted because 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 ABENDS, 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 ABENDS, the data set is deleted because of the conditional disposition of DELETE.

DLM Parameter

Keyword, Optional

The DLM parameter allows you to use a delimiter other than /* to terminate data defined in the input stream. By assigning a different delimiter in the DLM parameter, you can include a standard delimiter (/*) as data in the input stream.

For further information on the use of the DLM parameter, see the section titled *The Delimiter Statement* in this publication.

DLM=delimiter

delimiter

two characters that indicate the end of a group of data in the input stream.

Rules for Coding

1. The delimiter can be any two characters.
2. If the delimiter contains special characters, enclose it in apostrophes, (5-8 punch).
3. If you include an ampersand or an apostrophe in the delimiter, code each ampersand or apostrophe as two consecutive ampersands or apostrophes.
4. The DLM parameter has meaning on statements defining either data in the input stream or associated data on diskettes (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 delimiters override any delimiter implied by the DD * or DD DATA statement. You must terminate the data with the characters you assigned in the DLM parameter.
6. *Error processing:* If the system encounters an error on the DD statement before the DLM parameter, it will not recognize the value assigned as a delimiter. An EOF on the input reader device or an end-of-data set for associated data on diskette will also cause the system to end an input data set.

Examples of the DLM Parameter

```
//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.

DSID Parameter

Keyword, Optional

The DSID parameter specifies the data set identifier of an input or output data set on diskette for the IBM 3540. For additional information see *OS/VS1 IBM 3540 Programmer's Reference*, GC24-5110.

DSID=(id[,V])

id

specifies the data set identifier. The identifier must be one to eight characters. The characters must be alphameric, national, minus (hyphen), or left brace. The first character must be alphabetic or national.

,V

specifies that the data set must be verified before processing (input only).

Rules for Coding

1. The DSID keyword results in a JCL error if not on a DD *, DD DATA, or DD SYSOUT statement.
2. If you code only the id, you can omit the parentheses.
3. DSID on the DD * or DD DATA statement is interpreted, but used only when the JCL is processed by a system reader specifying associated data set processing.
4. Along with DSID, you can specify volume serial and logical record length information on the DD * and DD DATA statements.
5. DSID is mutually exclusive with DDNAME, MSVGP, and user-written writer name (program name).

Example of the DSID Parameter

```
//JOB1      JOB      , ,MSGLEVEL=(1,1)
//STEP      EXEC     PGM=AION
//SYSIN     DD       *,DSID=(ABLE,V),VOL=SER=123456,
//          DCB=LRECL=80
//OUTPUT    DD       SYSOUT=E,DCB=LRECL=128,DSID=BAKER
```

In this example the input is found on diskette in data set ABLE and must be verified. The output will be on diskette in data set BAKER.

DSNAME Parameter

Keyword, Optional

The DSNAME parameter assigns a name to the data set. The system uses the data set name to locate the data set on the volume.

For further information on identifying the data set to the system, see Appendix A of this publication.

$$\left. \begin{array}{l} \{ \text{DSNAME} \} \\ \{ \text{DSN} \} \end{array} \right\} = \left(\begin{array}{l} \text{dsname} \\ \text{dsname(member name)} \\ \text{dsname(generation number)} \\ \text{dsname(area name)} \\ \&\&\text{dsname} \\ \&\&\text{dsname(member name)} \\ \&\&\text{dsname(area name)} \\ \text{*ddname} \\ \text{*stepname.ddname} \\ \text{*stepname.procstepname.ddname} \end{array} \right)$$

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 (zero or a signed integer.)

dsname(area name)

identifies a nontemporary indexed sequential data set name and an area of that data set (INDEX, PRIME, or OVFLOW).

εεdsname

specifies the name you want assigned to a temporary data set.

εεdsname(member name)

specifies the name you want assigned to a temporary partitioned data set and to a member within that data set.

εεdsname(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 one to eight 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 one through eight characters, excluding the ampersands; the first character following an ampersand 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 can omit the DSNAME parameter if the data set is created and deleted in the job, that is, if the data set is temporary.
4. The DSNAME, DSID, and DDNAME parameters are mutually exclusive parameters; therefore, when the DDNAME or DSID parameter is coded, do not code the DSNAME parameter.

Examples of the DSNAME Parameter

```
//DD1      DD      DSNAME=ALPHA,DISP=( ,KEEP ),  
//                          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.

```
//DD2      DD      DSNAME=PDS( PROG12 ),DISP=( OLD,KEEP ),  
//                          UNIT=2314,VOLUME=SER=882234
```

This DD statement retrieves a member of a partitioned data set named PDS.

```
//DD3      DD      DSNAME=εεWORK,UNIT=2400
```

This DD statement defines a temporary data set. Because the data set is to be deleted at the end of the job step, the DSNAME parameter can be omitted. However, it may be included to facilitate a later reference to a passed data set, for example, DSNAME=εεWORK,DISP=OLD.

```
//STEP1 EXEC PGM=CREATE
//DD4 DD DSNNAME=&&ISDATA( PRIME ),DISP=( ,PASS ),
// UNIT=( 2311,2 ),SPACE=( CYL,( 10,,2),,
// CONTIG ),VOLUME=SER=( 33489,33490 )
//STEP2 EXEC PGM=OPER
//DD5 DD DSNNAME=* .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. Because the temporary data set will be passed when it is defined in STEP1, STEP2 can retrieve the data set.

FCB Parameter

Keyword,Optional

The FCB parameter specifies the forms control image to be used to print an output data set on a 3203-4 printer, a 3211 printer, a 3525 card punch with the read feature, or the IBM 3800 Printing Subsystem.

For the 3800, use the FCB parameter to request a long-page dump of 8 lines per inch and 80 lines per page.

For further information on the forms control buffer, see *OS/VSI Data Management for System Programmers*, GC26-3837.

$$\text{FCB}=(\text{image-id} \left[\begin{array}{l} \text{,ALIGN} \\ \text{,VERIFY} \end{array} \right])$$

image-id

the code that identifies the image to be loaded into the forms control buffer. Use STD3 on a problem program SYSABEND or SYSUDUMP DD statement to request a long-page dump.

,ALIGN

requests the operator to check the alignment of the printer forms before the data set is printed (not supported for the 3800).

,VERIFY

requests the operator to verify that the image displayed on the printer is the desired one. The operator is also given an opportunity to align the printer forms.

Rules for Coding

1. The image-id can be one to four alphameric and national (#, @, \$) characters. The first character must be an alphameric or national character.
2. For the 3203-4 and the 3211, if you omit the FCB parameter, the default image is used if it is currently in the buffer. Otherwise, the operator is requested to specify an image. For the 3800, if you omit the FCB parameter, the default image specified in the writer procedure is used. If none is specified in the writer procedure, the hardware default is used.
3. The FCB parameter is ignored if the data set is not written to a 3203-4 printer, to a 3211 printer, to a 3525 card punch with the read feature, or to a 3800.
4. 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.
5. If you do not code ALIGN or VERIFY, you need not enclose the image-id in parentheses.

6. For the 3800, the ALIGN subparameter is syntax checked, but otherwise ignored.
7. High-Density Dump:

To request dump output with 80 lines per page at 8 lines per inch, you must specify FCB=STD3 on the dump-related DD statement.

You can also request dump output that has 204 characters per line by specifying CHARS=DUMP on the dump-related DD statement.

Examples of the FCB Parameter

```
//DD1 DD UNIT=3211,FCB=(IMG1,VERIFY)
```

This DD statement defines the output data set that is 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. Because 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.

```
//DD2 DD SYSOUT=A,FCB=IMG2
```

This DD statement defines an output data set that is written to the device that corresponds with class A. The FCB parameter is ignored if the device is not a 3211 printer or a 3800.

```
//SYSABEND DD SYSOUT=A,FCB=STD3,CHARS=DUMP
```

This DD statement specifies that the problem program's SYSABEND data set be written in long-page (8 lines per inch and 80 lines per page) and double-density (204 characters per line) format.

FLASH Parameter

Keyword, Optional

The FLASH parameter identifies the forms overlay frame to be inserted into the 3800 and the number of copies of a data set on which the overlay is to be flashed. For further information on the FLASH parameter, see *IBM 3800 Printing Subsystem Programmer's Guide*, GC26-3846.

FLASH=(overlay name[,count])

overlay name

a one-to-four character name (alphameric and national characters) identifying a forms overlay frame.

count

the number of copies that are to be flashed with the overlay, beginning with the first copy. The maximum value of count is 255.

Rules for Coding

1. You can code the FLASH parameter together with the SYSOUT or UNIT parameter.
2. Refer to Figure 9 for parameters that are mutually exclusive with FLASH.
3. You must not omit the forms overlay name.
4. The maximum number of copies flashed will be equal to, but not greater than, the specification in the COPIES parameter.
5. If you omit count, a default of 255 is used so that all copies printed are flashed.
6. If you omit count, you need not code the parentheses.
7. When you code FLASH together with the UNIT parameter, data management sets up the 3800 with the flash count on the FLASH parameter or uses the default of 255. The actual number of copies flashed depends on the program that uses the 3800 directly.

Examples of the FLASH Parameter

```
//DD1 DD SYSOUT=A,FLASH=( #ABC,10 )
```

This statement requests that the forms overlay frame #ABC be inserted into the 3800 and that the first ten copies of the data set printed be flashed with the forms overlay.

```
//DD2 DD SYSOUT=A,FLASH=ABC3
```

This statement requests that all copies of the data set printed on the 3800 SYSOUT device be flashed with the overlay ABC3.

HOLD Parameter

Keyword, Optional

The HOLD parameter specifies that an output data set is held on a queue until the central operator or the user to which the data is being sent issues a ROUTE command or a RELEASE command.

HOLD= { YES }
 { NO }

YES

specifies that processing of the output data set by a JES writer is deferred until a ROUTE or RELEASE command is issued.

NO

specifies that processing of the output data set is to proceed normally.

Rules for Coding

1. The HOLD parameter must be coded with the SYSOUT parameter on the DD statement.
2. *Default:* If you do not code the HOLD parameter, a default of HOLD=NO is assumed and the output data set is enqueued normally.
3. If you are not receiving the output at your own workstation, you should inform either the central operator or the workstation to which the output will be sent that a ROUTE or RELEASE command must be issued to process the output data set.

Example of the HOLD Parameter

```
//JOB01      JOB          , 'HAROLD DUQUETTE' ,MSGLEVEL=1  
//STEP01    EXEC        PGM=MJCOSCO  
//DD1       DD          SYSOUT=B,DEST=STAT04 ,HOLD=YES
```

The output from JOB01 is held on a queue until the user identified by STAT04 or the central operator issues a ROUTE command or RELEASE command.

LABEL Parameter

Keyword,Optional

The LABEL parameter:

- Describes the data set label associated with a data set.
- Describes the sequence number of a data set that does not reside in the first location on a reel.
- Assigns a retention period to a data set.
- Assigns password protection to a data set.
- Overrides the OPEN macro instruction for BSAM data sets.

For further information on tape label definitions and processing, see *OS/VS Tape Labels, GC26-3795*.

For more information on using the LABEL parameter, see Appendix A of this publication.

```
LABEL=([data set sequence number] [,SL  
,SUL  
,AL  
,AUL  
,NSL  
,NL  
,BLP  
,LTM  
, [,PASSWORD [,IN [,EXPDT=yyddd  
,NOPWREAD ,OUT ,RETPD=nnnn ])
```

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.

,AUL

specifies that the data set has American National Standard user labels.

,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 data set may have a leading tapemark.

- ' 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 supplies the system with the correct password; that is, 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.
- ' specifies that another subparameter follows and, for a new data set, the data set will not be password protected.
- ,IN**
specifies that the data set is processed for input only.
- ,OUT**
specifies that the data set is processed for output only.
- ,EXPDT=yyddd**
specifies the date when the data set can be deleted or overwritten by another data set. Assign a two-digit year number and a three-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 code one subparameter and have omitted a previous 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. Data sets whose expiration date is the current date are considered to have an expired date and the data sets can be deleted or overwritten by other data sets.
4. To make sure that a temporary data set is deleted at the end of the job, you should not specify a retention period or an 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 an expiration date for a temporary data set, the system will not delete the data set until the time period has expired.

5. If the data set has IBM standard labels, you can omit the subparameter SL.
6. 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.
7. If you are processing ASCII data on unlabeled (NL) tapes, you must code OPTCD=Q in your DCB macro instruction or on your DD statement.
8. The LABEL, DDNAME, and SYSOUT parameters are mutually exclusive parameters; therefore, if DDNAME or SYSOUT is coded, do not code the LABEL parameter.
9. You must code the LABEL parameter 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.
 - 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).
10. If BLP is specified for a tape volume with the standard labels, the system treats anything within tapemarks as a data set. In order for the tape to be positioned properly, the sequence number must reflect all header and trailer labels and data sets that precede the desired data set.

Examples of the LABEL Parameter

```
//DD1      DD      DSNAME=HERBI,DISP=(NEW,KEEP),
//          UNIT=TAPE,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 tape volume has nonstandard labels; (3) this data set is to be kept for 188 days.

```
//DD2      DD      DSNAME=A.B.C,DISP=(,CATLG,DELETE),
//          UNIT=2400-2,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).

```
//DD3      DD      DSNAME=SPECS,UNIT=2400,  
//          VOLUME=SER=10222,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.

```
//STEP1    EXEC    PGM=FIV  
//DDX      DD      DSNAME=CLEAR,DISP=(OLD,PASS),  
//          UNIT=2400-4,VOLUME=SER=1257,  
//          LABEL=(,NSL)  
//STEP2    EXEC    PGM=BOS  
//DDY      DD      DSNAME=*.STEP1.DDX,DISP=OLD,  
//          LABEL=(,NSL)
```

The DD statement that named DDX in STEP1 defines an existing data set that has nonstandard labels and requests that the system pass the data set. The DD statement named DDY in STEP2 receives the passed data set. Unit and volume information is not specified, because this information is available to the system. The label type is not available to the system and must be coded.

MODIFY Parameter

Keyword, Optional

Use the MODIFY parameter to identify the name of a copy modification module for the 3800 Printing Subsystem. The data in the module replaces the variable data that would otherwise print in the positions specified.

For further information on the use of the MODIFY parameter, see *IBM 3800 Printing Subsystem Programmer's Guide*, GC26-3846.

MODIFY=(module name[,table reference character])

module name

the name of a copy modification module that resides in SYS1.IMAGELIB.

table reference character

is used to select one of the character arrangement tables for the translation of the copy modification data. The table reference character can assume a value of 0, 1, 2, or 3 and corresponds to the order in which you specify character arrangement tables in the CHARS parameter. The absence of the table reference character results in the default value of zero and causes the first character arrangement to be selected.

Rules for Coding

1. You must not omit the module name subparameter.
2. Refer to Figure 9 for parameters that are mutually exclusive with MODIFY.
3. The table reference character value, if greater than zero, must have a corresponding character arrangement table specified in the CHARS parameter.
4. If you omit the table reference character, you need not code the parentheses.

Examples for the MODIFY parameter

```
//DD1      DD      SYSOUT=A,CHARS=(GS15,GS10),  
//          MODIFY=(DIST,1)
```

This DD statement requests the character arrangement tables GS15 and GS10 be used in printing a SYSOUT data set on the 3800 and the copy modification data named DIST be translated against the table GS10.

```
//DD2      DD      UNIT=3800,MODIFY=DIST,CHARS=GS10
```

This DD statement specifies that the copy modification data named DIST should be translated against the GS10 character arrangement table.

MSVGP Parameter

Keyword, Optional

The MSVGP parameter enables you to specify the identifier of a group of mass storage volumes that reside on a Mass Storage System (MSS) device. For more information see *OS/VS Mass Storage System (MSS) Planning Guide*, GC35-0011.

MSVGP=(id|,ddname)

id

specifies MSVGP identification with one to eight alphanumeric or national characters in any order.

ddname

specifies the Master In DD statement from which the Master Out data set is to be separated.

Rules for Coding

1. MSVGP is mutually exclusive with VOL=SER, SYSOUT, DDNAME, VOL=REF, SUBALLOC, and DSID. If MSVGP is coded with DISP=OLD or DISP=SHR, the MSVGP information is ignored.
2. Only one ddname can be coded and it must refer to a previous DD statement in the same step.
3. If only id is coded, you may omit the parentheses.
4. If you code ddname, you must code id.
5. If volume separation within the group is not possible, the job is failed.
6. For concatenated DDs (including GDG ALLs), ddname refers only to the first DD statement of the concatenation.
7. To extend multivolume data sets to a nonmounted volume, the unit count in the UNIT parameter must be less than the volume count in the VOLUME parameter. Either the MSVGP parameter or the VOL=PRIVATE parameter must be specified guaranteeing allocation of a nonsharable unit.
8. For a new nonspecific permanent data set request where MSVGP is not specified, a mounted 3330V storage volume is used if one exists. If one does not exist, a volume is selected from SYSGROUP.
9. To guarantee allocation to SYSGROUP for nonspecific requests, specify MSVGP=SYSGROUP or VOLUME=PRIVATE, in which case the SPACE parameter is required.

Example of the MSVGP Parameter

```
//DD1      DD      UNIT=3330V,MSVGP=A
```

This example mounts a volume from group A.

```
//DD1      DD      DSN=MASTER,DISP=OLD  
//DD2      DD      DSN=MASTROUT,UNIT=3330V,DISP=(,PASS),  
//          MSVGP=(LR$6143@,DD1)
```

DD1 specifies an existing cataloged data set. DD2 describes a new data set that will be allocated to a volume that belongs to the mass storage volume group LR\$6143@. Since DD1 is specified as the ddname in the MSVGP parameter coded on DD2, the system ensures that the DD2 data set, MASTROUT, is not allocated to the same volume(s) that contain the DD1 data set MASTERIN.

OUTLIM Parameter

Keyword,Optional

The OUTLIM parameter specifies a limit for the number of logical records you want included in the output data set being routed through the output stream. When the limit is reached, an exit is taken to a user supplied routine that determines whether to cancel the job or to increase the limit. If the exit routine is not supplied, the job is canceled.

OUTLIM=number

number

the maximum number of logical records to be included in the output data set being routed through the output stream. The largest number that can be specified is 16,777,215.

Rules for Coding

1. The OUTLIM parameter is ignored unless SYSOUT is coded in the operand field of the same DD statement.
2. The value specified for OUTLIM can be any number from 1 through 16,777,215.
3. If you omit the OUTLIM parameter on a SYSABEND or SYSUDUMP DD statement, or if you code OUTLIM=0, no output limiting is done. If you omit it on any other DD statement, the output data set is limited by the default value specified at system generation time.
4. **Determining a 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.)
5. The OUTLIM and DDNAME parameters are mutually exclusive. Do not code them together on one DD statement.

Example of the OUTLIM Parameter

```
//OUTPUT DD SYSOUT=F,OUTLIM=1000
```

The limit for the number of logical records is 1,000.

QNAME Parameter

Keyword,Optional

The QNAME parameter enables a user to access messages received through TCAM for processing by an application program.

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.

Rules for Coding

1. The process name must consist of one through eight alphanumeric or national (#, \$, @) characters. The first character must be an alphabetic or national character.
2. The process name must be 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.

Example of the QNAME Parameter

```
//DYD      DD      QNAME=FIRST,DCB=(RECFM=F,  
//          LRECL=80,BLKSIZE=320)
```

This DD statement is used in an application program to define data that will 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 will be routed. The DCB parameter supplies information that was not supplied in the DCB macro instruction for the DCB.

SEP Parameter

Keyword,Optional

The SEP parameter requests channel separation for specific data sets defined in a job step. 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.

SEP=(ddname[,...])

ddname

the names of up to eight earlier DD statements in the same job step that define data sets from which channel separation is desired.

Rules for Coding

1. Separate ddnames by a comma.
2. If only one ddname is coded, you can omit the parentheses.
3. **Default:** If you code neither the SEP nor AFF parameter, any available channel, consistent with the UNIT parameter requirement, is assigned by the system.
4. If channel separation is critical, use the UNIT parameter to specify a particular channel, using an absolute address or group name.
5. 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.
6. Requests for channel separation are ignored for any data sets that have been allocated devices by the automatic volume recognition (AVR) option.
7. With the I/O load balancing option in use, the SEP parameter is not necessarily honored for new direct access data sets.

Examples of the SEP Parameter

```
//STEP1 EXEC PGM=STARTS
//DD1 DD DSNAME=X.Y.Z,DISP=OLD
//DD2 DD DSNAME=&&WORK,DISP=(,PASS),UNIT=2314,
// SPACE=(CYL,(3,1))
//DD3 DD DSNAME=NABS,DISP=OLD,VOLUME=SER=7110,
// UNIT=2314
//DD4 DD DSNAME=PARE,DISP=OLD,VOLUME=SER=E59,
// UNIT=2314,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. Because 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.

SPACE Parameter

Keyword,Optional

The SPACE parameter indicates how much space should be allocated on a direct access volume for a new data set.

For further information on the SPACE parameter, see *OS/VSI JCL Services*, GC24-5100.

$$\left\{ \begin{array}{l} \text{SPACE}=(\left\{ \begin{array}{l} \text{TRK} \\ \text{CYL} \\ \text{block length} \end{array} \right\} ,(\text{primary quantity} [\text{,secondary quantity}] [\text{,directory} \\ \text{,index}]) [\text{,RLSE}] [\text{,CONTIG} \\ \text{,MXIG} \\ \text{,ALX}] [\text{,ROUND}]) \\ \text{SPACE}=(\text{ABSTR},(\text{primary quantity},\text{address} [\text{,directory} \\ \text{,index}])) \end{array} \right\}$$

TRK

specifies 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 additional space is required, or how many more blocks of data may be included if additional space is required. This secondary allocation can be done 15 times.

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 how many cylinders are required for the index of an indexed sequential data set.

,RLSE

specifies the release of space allocated to the data set that is not used when the data set is closed.

specifies that allocated space that is not used is not to be released and that 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 that the ROUND subparameter follows.

,ROUND

specifies that space is requested by specifying the average block length of the data, and the space allocated to the data set must be equal to an integral number of 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 relative 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 you assign a data set to a device class that contains both direct access devices and tape devices, for example, UNIT=SYSSQ, you should code the SPACE parameter.
2. If you do not code secondary, directory, or index quantities, you need not enclose the primary quantity in parentheses.
3. 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.
4. Code the second format of the SPACE parameter when you want a data set placed in a specific position on a direct access volume.

5. **Unit of measurement:** 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.

6. **Primary quantity:** Enough available space must be on one volume to satisfy the primary quantity. If you request that a particular volume be used and not enough space is available on the volume to satisfy your request, the job step is terminated.

For indexed sequential data sets, if you specify a number of tracks, that number must be equal to one or more cylinders. 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 terminates.

For indexed sequential data sets, the relative track number (address) must correspond to the first track on a cylinder. Capacities of a number of direct access devices are listed in Figure 11.

7. **Secondary quantity:** The system computes the number of tracks required for the secondary quantity based on what is specified in the DCB subparameter BLKSIZE. Therefore, include BLKSIZE on the DD statement.

If you 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) not enough space is 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 that this be done 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 nonspecific volume request, code PRIVATE and specify more than one volume in the VOLUME parameter.

For a non-specific request, if no space is available on the current volume, and at least one volume is demountable, the system requests scratch volumes to be mounted until either the data set is complete or 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 ABEND.

For a specific request, if no space is available on the current or next volume specified, the job step will abend. (Note: The search for space is only done on current and current+1 volumes.)

8. **Directory:** If you are creating a partitioned data set, you must request space for a directory.

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.

9. **RLSE:** If you specify RLSE and an ABEND occurs, unused space is not released.

The RLSE subparameter is ignored when the TYPE=T option is coded in the CLOSE macro instruction.

10. **MXIG or ALX:** Do not code either the MXIG or ALX subparameter for an indexed sequential data set.

11. **Mass Storage System:** You must code the SPACE parameter when VOL=SER is coded. When you code MSVGP, the SPACE parameter is optional.

If you do not code MSVGP or VOL=SER, you must code SPACE whether or not you code VOL=PRIVATE.

If you code MSVGP and do not code SPACE, the default is CONTIG. If you want noncontiguous primary space allocation, you must specify SPACE.

Examples of the SPACE Parameter

```
//DD1      DD      DSNAME=εεTEMP,UNIT=MIXED,  
//          SPACE=(CYL,10)
```

This DD statement defines a temporary data set and requests that the system 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 (that is, the type of units and a primary quantity), and requests the system to allocate ten cylinders.

```
//DD2      DD      DSNAME=ELLN,DISP=(,KEEP),UNIT=2314,  
//          VOLUME=SER=11257,SPACE=(1024,  
//          (100,25),,,ROUND),DCB=BLKSIZE=2048
```

This DD statement defines a new data set that is written on a direct access volume. The SPACE parameter requests that the system compute the space required for the primary quantity. The system computes the space required based on an average block length of 1,024 bytes, and up to 100 blocks of data are written. If more space is required, the system computes how much additional space is to be allocated. The system computes the space

required based on a maximum block length of 2,048 bytes (specified in the BLKSIZE subparameter), and up to 25 blocks of data are written. Because 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.

```
//DD3      DD      DSNAME=PDS12,DISP=(,KEEP),UNIT=2314,  
//          VOLUME=SER=26143,SPACE=(TRK,  
//          (200,,10),,CONTIG)
```

This DD statement defines a new partitioned data set. The system allocates 200 tracks to the data set and ten 256-byte records for a directory. Because the CONTIG subparameter is coded, the system allocates 200 contiguous tracks on the volume.

```
//DD4      DD      DSNAME=INDSEQ(INDEX),UNIT=2314,  
//          DCB=DSORG=IS,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 one cylinder), beginning with the fortieth track on the volume (the fortieth track on the volume is the beginning of the third cylinder).

SPLIT Parameter

Keyword,Optional

The SPLIT parameter allocates space to two or more new data sets that are to share cylinders.

$$\text{SPLIT} = \left\{ \begin{array}{l} (n, \text{CYL}, (\text{primary quantity}, \text{secondary quantity})) \\ n \\ (\text{percent}, \text{block length}, (\text{primary quantity}, \text{secondary quantity})) \\ \text{percent} \end{array} \right\}$$

n
is 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
is the number of tracks per cylinder you want allocated to the data set.

percent
is the percentage of tracks per cylinder you want allocated to the first data set – a number from 1 through 99. When calculating the actual number of tracks to be allocated the system rounds down to the next full track. If the percentage is less than one track, it causes a JCL error.

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
is the percentage of tracks per cylinder you want allocated to the data set defined on the DD statement. When calculating the actual number of tracks to be allocated the system rounds down to the next full track. If the percentage is less than one track, it causes a JCL error.

Rules for Coding

1. Do not code the SPLIT parameter for direct, partitioned, and indexed sequential data sets.
2. 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.
3. If you use the SPLIT parameter 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.
4. 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.
5. The first DD statement that contains the SPLIT parameter must contain volume and unit information, and may contain the MSVGP parameter. You need not code volume and unit information on the following DD statements that contain the SPLIT parameter, and you cannot code MSVGP on these statements.
6. If you do not specify a secondary quantity, you need not enclose the primary quantity in parentheses.
7. **Block length:** 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.

Examples of the SPLIT Parameter

```
//STEP1      EXEC      PGM=CREATE
//DD1        DD         DSNAME=QUEST,DISP=( ,KEEP ),UNIT=2314,
//           VOLUME=SER=757500,SPLIT=( 7,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) seven 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 four tracks per cylinder to this data set. The DD statement named DD3 requests that the system allocate three tracks per cylinder to this data set.

```
//STEP2      EXEC      PGM=PAGE
//DDX        DD         DSNAME=ISSA,DISP=( ,KEEP ),UNIT=2314,
//           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 parame-

ter: (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 1,024 bytes and 700 blocks are required. The DD statement named DDY requests the system to 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. Because the first DD statement in the sequence does not specify a secondary quantity, the job will ABEND if any of the data sets exceeds its allocated space.

SUBALLOC Parameter

Keyword,Optional

The SUBALLOC parameter places a series of new data sets in a sequence in one area of contiguous space on a direct access device.

SUBALLOC=({ TRK
CYL
block length } ,primary quantity [,secondary quantity] [,directory]) { ,ddname
,stepname.ddname
,stepname.procstepname.ddname })

TRK

specifies 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 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 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, MSVGP, DDNAME, and SYSOUT parameters are mutually exclusive parameters; therefore, when SPACE, SPLIT, MSVGP, DDNAME, or SYSOUT is coded, do not code the SUBALLOC parameter.
4. Do not use the SUBALLOC parameter to allocate space for an indexed sequential data set.
5. When you delete a suballocated data set, the allocated space is released, but is not returned to the master data set.

Examples of the SUBALLOC Parameter

```
//STEP1 EXEC PGM=PREP
//DD1 DD DSN=DUM,DISP=(,KEEP),UNIT=2305-2,
// VOLUME=SER=ALLDS,SPACE=(CYL,
// 50,,CONTIG)
//STEP2 EXEC PGM=BSPED
//DD2 DD DSN=SPEC50,DISP=(,KEEP),
// SUBALLOC=(CYL,(20,1),STEP1.DD1)
//DD3 DD DSN=SPEC51,DISP=(,KEEP),
// SUBALLOC=(TRK,(44,7),STEP1.DD1)
//DD4 DD DSN=SPEC52,DISP=(,KEEP),
// SUBALLOC=(CYL,25,STEP1.DD1)
```

The data set from which space is suballocated is defined on the DD statement named DD1 in STEP1. Fifty contiguous cylinders will be allocated to the data set. 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. Because of the request for suballocation, the order of the data sets on the volume will be DUM, SPEC50, SPEC51, and SPEC52.

```
//STEPX EXEC PGM=GARV
//DD5 DD DSN=SIMP,DISP=(,KEEP),UNIT=2314,
// VOLUME=SER=315046,
// SPACE=(CYL,100,,CONTIG)
//DD6 DD DSN=FIELD,DISP=(,KEEP),
// SUBALLOC=(1024,(800,60),DD5)
//STEPY EXEC PGM=BERSS
//DD7 DD DSN=PDS,DISP=(,KEEP),
// SUBALLOC=(CYL,(75,,8),STEPX.DD5)
```

The data set from which space is suballocated is defined on the DD statement named DD5 in STEPX. One hundred contiguous cylinders will be allocated to the data set. 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 that the system allocate eight 256-byte records for a directory.

SUBSYS Parameter

Keyword, Optional

The SUBSYS parameter provides a means for defining a generalized subsystem data set and for passing subsystem dependent data to a subsystem.

SUBSYS=(name [,parm]...)

name

specifies the one- to four-character name of the subsystem. The characters can be alphabetic, numeric, or national. The first character must be alphabetic or national.

parm

specifies a character string whose meaning is defined by the subsystem. Each character string can consist of from 1 to 67 characters. The maximum number of character strings is 253.

Rules for Coding

1. If a character string or group of character strings contains special characters, apostrophes must be used to enclose each string or group. A character string enclosed in apostrophes must not be continued on another line.
2. If only the subsystem name is specified, the parentheses are not needed.
3. Checkpoint/restart is not supported for job steps containing DD statements specifying SUBSYS. The action of the CHKPT macro is suppressed for such job steps.
4. Only the DCB parameter may be coded on the same DD statement as SUBSYS. All other parameters are mutually exclusive with SUBSYS.

Example of the SUBSYS Parameter

```
//LABEL          DD      SUBSYS=( SUB1, PH, 'P2=PARAM2',  
//              'P3=PARAM3',  
//              CPQ )
```

This DD statement defines the data set as one to be processed by subsystem SUB1. PH, P2=PARAM2, P3=PARAM3 and CPQ are subsystem dependent data that is passed to subsystem SUB1 when the job containing this DD statement is executed.

SYSOUT Parameter

Keyword, Optional

The SYSOUT parameter assigns an output class to an output data set.

For further information on the SYSOUT parameter, see *OS/VSI JCL Services*, GC24-5100.

SYSOUT= { classname
(classname [,program name] [,form number] [,PROFILE='sysout profile string'])
PROFILE='sysout profile string' }

classname

specifies the class associated with the output device of your output data set written.

,program name

specifies the member name of a program in the system library that writes your output data set to a unit record device, instead of the system output writer. This field is mutually exclusive with DSID.

,

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.

PROFILE='sysout profile string'

the profile string should describe the requirements of your job. The format of the string is established by the system programmer for your installation.

Rules for Coding

1. The classname can be any alphanumeric character (A-Z, 0-9).
2. The form number is one to four alphanumeric and national (@, \$, #) characters.
3. The OUTLIM, UCS, FCB, COPIES, BURST, FLASH, CHARS, MODIFY, COMPACT, DEST, HOLD, DSID 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.
4. The DISP, DDNAME, AFF, SEP, VOLUME, LABEL, SPLIT, MSVGP, 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.
5. When PROFILE='profile string' is coded, it overrides the SYSOUT classname.

6. If PROFILE= is the only SYSOUT subparameter coded, it does not have to be enclosed in parentheses. However, if the sysout profile string continues from one line to another, each line must begin and end with an apostrophe and the entire group must be enclosed in parentheses.

Examples of the SYSOUT Parameter

```
//DD1      DD      SYSOUT=P
```

This DD statement specifies that the data set is written to the device corresponding to class P.

```
//JOB50    JOB      , 'C. BROWN' ,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 written to the device corresponding to class C. The DCB parameter is coded to complete the DCB associated with this data set. Because 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.

```
//DD5      DD      SYSOUT=( F , , 7402 )
```

This DD statement specifies that the data set is written to the device corresponding to class F and the output data set is printed on a special form. The form number is 7402.

```
//DD8      DD      SYSOUT=PROFILE='FORM=SINGLE'
```

The user has specified his job's requirements and the system assigns the output class based on this information when compared with the table created for the installation by the system programmer.

```
//LAQ      DD      SYSOUT=PROFILE=( 'SPACE=99' ,
//          'SIZE=99,CLASS=A,COMPILE=PL1' ,
//          'CHARACTER=A' )
```

This example shows the method of continuing the PROFILE= subparameter of the SYSOUT parameter.

TERM Parameter (for RES)

Keyword, Optional

RES (remote entry services) provides the facility to submit jobs to a central computing center from a remote workstation and to route output to remote workstations.

The TERM parameter indicates to the system the presence of an RTAM (Remote Terminal Access Method) device, used with RES.

For further information on RES, see *OS/VSI RES System Programmer's Guide*, GC28-6878.

TERM=RT

RT

indicates that a remote unit record device is in use for RTAM and that the usual allocation processing is to be bypassed.

Rules for Coding

1. You can code TERM=RT only on a DD statement for a job that is a system task. The DD statement containing the TERM parameter is processed in the same way that requests for SYSIN, SYSOUT, and DUMMY are handled.
2. Do not code the TERM parameter on DD *, DD DATA, and SYSOUT DD statements.
3. Code the UNIT parameter with the TERM parameter to specify a specific unit record device.

Example of the TERM Parameter

```
//JOB01      JOB      'WOODLAND AND COSCO',MSGLEVEL=1
//IEFPROC    EXEC     PGM=IEFOSCO1,PARM='PA'
//IEFRDER    DD       UNIT=PR1,TERM=RT
```

UCS Parameter

Keyword, Optional

The UCS parameter describes the character set to be used for printing an output data set on a 1403, 3203-4, or 3211 printer.

For further information on the UCS parameter, see *OS/VS1 Data Management for System Programmers*, GC26-3837.

UCS=(character set code [,FOLD] [,VERIFY])

character set code

up to four characters that identify 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. The fold mode is described in the publication *IBM 2821 Control Unit*, GA24-3312. The fold mode is most often requested when all data is to be printed only in uppercase.

,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. 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.
2. **Default:** If you omit the UCS parameter and the data set is written to a printer with the UCS feature, a default character set established by the installation 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.

If you code the UCS parameter and the data set is not written to a printer with the UCS feature, the UCS parameter is ignored.

3. If you omit the FOLD and VERIFY subparameters, you need not enclose the character set code in parentheses.
4. 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.

5. The UCS and DDNAME parameters and the DCB subparameters RKP and CYLOFL are mutually exclusive parameters; therefore, if you code the DDNAME parameter or one of the DCB subparameters RKP or CYLOFL, do not code the UCS parameter.
6. The UCS parameter is ignored for a spooled output data set that is directed to a 1403 printer.
7. For spooled output data sets that are directed to a 3800, the UCS parameter is used if you don't specify the CHARS parameter for that data set.

Examples of the UCS Parameter

```
//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. Because VERIFY is coded, the character set image is displayed on the printer before the data set is printed.

UNIT Parameter

Keyword, Optional

The UNIT parameter specifies what types of devices and how many devices you want assigned to a data set.

For further information on use of the UNIT parameter, see *OS/VSI JCL Services*, GC24-5100.

$\left\{ \begin{array}{l} \text{UNIT}=(\left[\begin{array}{l} \text{unit address} \\ \text{device type} \\ \text{group name} \end{array} \right] \left[\begin{array}{l} \text{,unit count} \\ \text{,P} \\ \text{,} \end{array} \right] \text{[,DEFER][,SEP=(ddname,...)]}) \\ \text{UNIT=AFF=ddname} \\ \text{UNIT=symbolic address} \end{array} \right\}$

Note: This use of the UNIT parameter is restricted to remote devices only.

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. For a list of acceptable device types see Appendix C.

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 how many 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 you do not code the DEFER subparameter but do code the SEP parameter, 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 unit separation is desired. This data set will be assigned a different direct access device than the devices assigned to the data sets specified by the ddnames that follow. With the I/O Load Balancing option in use, the SEP subparameter is not necessarily honored for direct access data sets.

(ddname,...)

indicates 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 unit affinity is desired. The system assigns this data set to the same device(s) assigned to the data set specified by the ddname that follows, provided that the volumes are removable.

ddname

indicates the name of an earlier DD statement in the job step that defines a data set with which you want unit affinity.

symbolic address (remote unit devices only)

identifies a particular remote unit device by its symbolic type and a unit number.

Rules for Coding

1. If the only subparameter you code in the UNIT parameter is the first subparameter, you can omit the parentheses.
2. The UNIT and DDNAME parameters are mutually exclusive parameters; therefore, if you code DDNAME, do not code the UNIT parameter.
3. Do 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.

If you request a telecommunications device, other than a 3704, 3705, or 3791, the system allocates that device on a shared basis whether or not the device is already allocated or is online.

4. **Group name:** A group name is one through eight alphanumeric characters and can identify a device or a group of devices. The group can consist of devices of the same type or different direct access and tape device types.

When you code a group name, the system assigns any available device from the group. If a group consists of only one device, the system assigns that device.

If a data set that was created using the group name subparameter is to be extended, additional units allocated to it will be of the same type as specified in the original group name. However, the units allocated to the data set are not necessarily of that same group.

A group name for a group containing different device types that cannot process each other's volumes (for example, different types of DASD or 7-track and 9-track tape drives) should not normally be used with a unit count greater than 1. This is because different device types may be allocated and the system assumes any volume associated with a DD statement can be mounted on any device allocated to it. When you specify a disposition of catalog, the device type of the first device is placed in the catalog.

When the AVR (automatic volume recognition) feature is included in the system and you specify a group name, this feature assigns devices to volumes already mounted, but does not request mounting of any volume that is not mounted.

5. **Default:** the unit count subparameter indicates how many devices you want assigned to a data set. If you omit the unit count subparameter or code 0, the system assigns one device, unless you have specified parallel mounting.

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 assigns one device, even if more devices were requested in an earlier DD statement.

Only in one case the system assigns more than one device — when two DD statements in a step request use of the same volume. If either of the two DD statements requests any other volume(s), the system assigns an additional device.

6. **Deferred mounting:** If you request deferred mounting of a volume and the data set on that volume is never opened by the processing program, the volume will not be 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.
7. **The SEP subparameter:** If you code the SEP subparameter, 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.
8. **Specific volume request:** 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 available devices, or if the request for unit separation should be ignored, or if the job should be canceled. 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. A message will not be issued in this case if unit separation cannot be satisfied.
9. The unit address and unit count subparameters are not mutually exclusive; however, if you code the unit address and unit count subparameters together, the unit count must be 1.
10. UNIT=AFF and DISP=NEW are mutually exclusive on the same DD statement for direct access devices. Therefore, if your data set is new, you cannot request unit affinity to another ddname.
11. UNIT=AFF to a SYSIN or SYSOUT DD statement is ignored.
12. **Mass Storage System:** If an old multivolume data set resides on volumes within a group, request parallel mounting or specify unit count equal to the number of volumes containing the data set.

To extend multivolume data sets to a nonmounted volume, unit count must be less than the volume count.

DEFER mounts should not be specified for volumes belonging to a MSVGP if there are new data set requests (DISP=NEW) in that job step using MSVGP from the same group.

Examples of the UNIT Parameter

```
//STEP2 EXEC PGM=POINT
//DDX DD DSNAME=EST,DISP=MOD,VOLUME=SER=
// (42569,42570),UNIT=(2314,2)
//DDY DD DSNAME=ERAS,DISP=OLD,UNIT=2400-2
//DDZ DD DSNAME=RECK,DISP=OLD,
// VOLUME=SER=(40653,13262),UNIT=AFF=DDX
```

The DD statement named DDZ requests the system to assign the same unit to this data set that it assigned to the data set defined on the statement named DDX. Because DDX requests two devices, these two devices are assigned to the data set defined on DDZ.

```
//DD1 DD DSNAME=AAG3,DISP=(,KEEP),
// 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.

```
//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.

```
//DD3 DD DSNAME=COLLECT,DISP=OLD,
// 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. Because DEFER is coded, the volume is not mounted until the data set is opened.

```
//STEP1 EXEC PGM=XTRA
//DDA DD UNIT=2314,SPACE=(1024,(150,20))
//DDB DD UNIT=2314,SPACE=(1024,(100,10))
//DDC DD UNIT=(2314,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.

VOLUME Parameter

Keyword, Optional

The VOLUME parameter identifies the volume(s) on which a data set resides or will reside.

For further information on the use of the VOLUME parameter, see *OS/VSI JCL Services*, GC24-5100.

$\left\{ \begin{array}{l} \text{VOLUME} \\ \text{VOL} \end{array} \right\} = \left(\left[\text{PRIVATE} \right] \left[\text{RETAIN} \right] \left[\text{.volume sequence number} \right] \left[\text{.volume count} \right] \left[\begin{array}{l} \text{,SER}=(\text{serial number},\dots) \\ \text{,REF}=\text{dsname} \\ \text{,REF}=\text{*}.\text{ddname} \\ \text{,REF}=\text{*}.\text{stepname}.\text{ddname} \\ \text{,REF}=\text{*}.\text{stepname}.\text{procstepname}.\text{ddname} \end{array} \right] \right)$

PRIVATE

indicates that no output data set can be allocated to this volume unless the volume is specifically requested, and that the volume is to be de-mounted after its last use in the job step unless RETAIN is coded or the data set is passed.

,RETAIN

indicates that for tape volumes, 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.

, indicates that the RETAIN subparameter is omitted and the volume sequence number or volume count subparameter follows.

,volume sequence number

specifies which volume of an existing multivolume data set should be used to begin processing.

, indicates processing of an existing multivolume data set begins 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 the volume count is omitted and either the SER or REF subparameter follows.

,SER=

indicates that serial numbers of the volumes on which the data set resides or will reside, are specified.

(serial number,...)

indicates 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`

is 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 that 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`, `DDNAME`, and `SYSOUT` parameters are mutually exclusive parameters; therefore, if you code `DDNAME` or `SYSOUT`, do not code the `VOLUME` parameter.
2. **Specific request:** When you make a specific volume request, that is, specify the volume's serial number, 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.
3. **Nonspecific request:** To make a nonspecific volume request, 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 to make a nonspecific volume request.
4. **The `PRIVATE` subparameter:** If you code only `PRIVATE`, you can omit the parentheses.

When you do not code `PRIVATE`, and you code the volume sequence number or volume count subparameter, you must code a comma to indicate the absence of `PRIVATE`.

5. **The `RETAIN` subparameter:** You do not need to code the `RETAIN` subparameter when the data set is to be passed; the system automatically retains the volumes on which the data set resides.

If you do not code `RETAIN` and you code the volume sequence number or volume count subparameter, you must code a comma to indicate the absence of `RETAIN`.

6. **The volume sequence number:** The volume sequence number must be less than or equal to the number of volumes on which the data set

exists; it can be up to four digits, and can range from 1 to 255.

Normally, you code a volume sequence number when you have not specified volume serial numbers on the DD statement (that is, you are retrieving a cataloged data set or you have coded a reference to an earlier DD statement or data set). If you code both a volume sequence number and a volume serial number(s) in the VOLUME parameter, the system begins processing with the volume that corresponds with the volume sequence number.

For a direct access device with DISP=MOD, the volume sequence number, when used, must point to the last volume of the existing data set.

7. **The volume count:** The volume count value can range from 1 to 255 on a DD statement and a maximum of 4095 volumes may be specified for a single job step.

When you make a nonspecific volume request and the data set may exceed one volume, request more than one volume in the volume count and code PRIVATE, or request the same number of devices as volumes.

When you request a nonspecific tape volume for a data set with no labels, if the volume(s) you initially specified is demountable, the system requests scratch volumes to be mounted until either the data set is complete or until all entries in the JFCB are filled. If the JFCB entries are already filled, or the volume is not demountable, the job step will ABEND. If you have specified a volume count greater than 99, duplicate volume serial numbers are assigned.

When you make a specific volume request and the data set may require 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 ensure that the data set can be written on more than one volume if it exceeds one volume.

If the data set requires more volumes than you specify in the volume count, the system requests that as many as 14 additional volumes be mounted according to the formula:

$$\text{maximum number of volumes mounted} = 15 \times \left[\frac{(n+9)}{15} \right] + 5$$

$\left[\frac{(n+9)}{15} \right]$ denotes the greatest possible integer, dropping any remainder.

n is the greater of volume count and specific volume requests (explicit or implicit).

8. **The volume serial number:** You can specify a maximum of 255 volume serial numbers per DD statement and a maximum of 4,095 volume serial numbers per job step.

A volume serial number must be one to six characters in length. If the number is less than six characters, it is padded with trailing blanks. It can contain any alphanumeric 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 some typewriter heads or printer chains, difficulties in volume serial recognition may arise if you use other than alphameric characters.

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.

Do not use SCRCH or Lnnnnn as a volume serial number. For optical readers, if no volume serial number is specified, VOLUME=SER=OCRINP is assumed.

9. Each volume in an installation must have a unique serial number regardless of the volume type (that is, tape, disk, diskette).
10. When the same volume serial number is on more than one DD statement within a job step, the UNIT parameter is checked only on the first DD statement.
11. **Backward References:** 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, 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 cannot contain special characters, except for periods used in a qualified name. References to SYSIN (DD * or DD DATA) or SYSOUT DD statements are ignored.

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, 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.

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.

If the ddname is used with REF=, the label type is also copied from the referenced DD statement.

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.

12. **Mass Storage System:** VOL=SER and VOL=REF are mutually exclusive with MSVGP.

When you use VOL=SER, the SPACE parameter is required. When you use the MSVGP parameter, SPACE is optional.

To guarantee allocation to SYSGROUP for a nonspecific request, specify VOLUME=PRIVATE or MSVGP=SYSGROUP.

Examples of the VOLUME Parameter

```
//DD1      DD      DSNAME=STEP,UNIT=2314,DISP=OLD  
//          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. Because PRIVATE is coded in the VOLUME parameter, the system does 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.

```
//DDB      DD      DSNAME=COMM,DISP=(NEW,KEEP),  
//          SPACE=(CYL,(30,2)),  
//          VOLUME=(PRIVATE,,,2),UNIT=2314
```

The DD statement named DDB defines a new data set for which the system is to assign a volume. Because only one device is requested (UNIT=2314) and the volume count is 2, PRIVATE is coded to ensure that the additional volume can be mounted if required.

```
DD2      DD      DSNAME=QUET,DISP=(MOD,  
//          KEEP),UNIT=(2400,2),  
//          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 four volumes be used to process the data set if required.

```
//DD3      DD      DSNAME=QOUT,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.

Satisfying Specific Volume Requests

In the following cases the system satisfies 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 being used by a concurrently executing step and is a removable volume that has not been assigned the nonsharable attribute. (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, if the request is for a permanent data set or PRIVATE is coded.

Satisfying Nonspecific Volume Requests

You can request four types of nonspecific volume requests:

1. A private volume for a temporary data set.
2. A private volume for a nontemporary data set.
3. A public volume for a temporary data set.
4. A storage volume for a nontemporary data set.

The system satisfies these different types of requests as follows. Because the system satisfies the first two types of requests in the same way, they 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.

Section V: The COMMAND Statement

Control Statement

The COMMAND statement specifies an operator command to be executed.

For further information on commands and for descriptions of their operands see *Operator's Library: OS/VS1 Reference*, GC38-0110 and *Operator's Library: OS/VS1 CRJE*, GC38-0335.

```
// command      operand  comments
```

The command statement consists of the characters // in columns 1 and 2, and three fields: operation (command), operand, comments.

These commands can be entered through the input stream:

BRDCST: Maintains information in the broadcast data set. (CRJE)

CANCEL: Immediately terminates the scheduling or execution of a job, cancels a job on the queue, or stops the writing of an output data set currently being processed by an output writer.

CENOUT: Causes output from remotely submitted jobs to be written locally. (CRJE)

DISPLAY: Causes a console display of certain system status information.

HOLD: Temporarily prevents one job, all jobs, or output from being selected for processing.

LISTBC: Lists the contents of the system broadcast data set.

LOG: Enters information into the system log.

LOGOFF: tops RTAM processing and also terminates all tasks initiated by user.

MODIFY: Changes the characteristics of a functioning output writer, a reader, or other tasks.

→ **MONITOR:** Allows monitoring of certain system activity.

MOUNT: Assigns 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.

MSG: Sends a message to a terminal user. (CRJE)

RELEASE: Enables the system to resume job selection, which had been suspended by the HOLD command, TYPRUN=HOLD on the JOB statement or TYPRUN=HOLD issued by a MODIFY command for the reader.

REPLY: Used to reply to messages from the system or from a process-

ing program that requests information.

RESET: Changes the class or priority of a job in an input or system output queue.

ROUTE: Redirects output for a specified job, user, or class.

SEND: Allows communication with remote users.

SET: Establishes the values of certain variables, such as the time of day and the date.

SHOW: Displays information pertaining to CRJE.

START: Starts a particular system process, for example, an input reader, graphic job processor, initiator.

STARTF: Starts a reader or writer.

STOP: Stops a system process that had been previously started by a START command or stops the console display effected by the DISPLAY command.

STOPMN: Cancels the MONITOR command.

UNLOAD: Removes the volume previously mounted in response to a MOUNT command.

USERID: Adds users to or deletes users from the system. (CRJE)

VARY: Places an I/O device or path into an online or offline status.

WRITELOG: Causes the system output writer to write the contents of the system log.

WRITER: Controls the output of a writer.

Rules for Coding

1. Follow the // in columns 1 and 2, with one or more blanks.
2. Follow the command with one or more blanks.
3. Code any required operands. Separate each operand with a comma.
4. Follow the operands with one or more blanks.
5. Code any comments.
6. The command statement cannot be continued.
7. A command statement may appear immediately before a JOB statement, an EXEC statement, a null statement, or another command statement.
8. If a command statement appears in the input stream between the boundaries of two jobs, the command is executed when the jobstream

is read. If it contains errors, the command is not executed and message IEF543I is issued.

9. If you include a command statement within a job, the command is executed when the job is selected for processing.
10. Disposition of commands read from an input stream is specified as a PARM parameter field in the cataloged procedure for the input reader.

Example of the Command Statement

```
//      START  INIT,,,AB START AN INITIATOR
```

This command starts an initiator. The characters A and B indicate that the initiator executes only jobs from classes A and B.

Section VI: The Comment Statement

Control Statement

The comment statement specifies a comment to be included in the output listing.

```
  /**comments
```

The comment statement consists of the characters `/**` in columns 1, 2, and 3, and the comments field.

Rules for Coding

1. Code the comments in columns 4 through 80.
2. You cannot continue comment statements using continuation conventions. If you cannot include all of the comments on one comment statement, code another comment statement.
3. The comment statement may appear anywhere after the JOB statement.
4. With the MSGLEVEL parameter, you can request an output listing of all the control statements processed in your job. You can identify comment statements by the appearance of `***` in columns 1, 2, and 3.

Example of the Comment Statement

```
/**THE COMMENT STATEMENT CANNOT BE CONTINUED,  
/**BUT IF YOU HAVE A LOT TO SAY, YOU CAN FOLLOW A  
/**COMMENT STATEMENT WITH MORE COMMENT  
/**STATEMENTS.
```

Section VII: The Delimiter Statement

Control Statement

The delimiter statement indicates the end of the data for a DD statement.

```
/*comments
```

The delimiter statement consists of the characters /* in columns 1 and 2, and the comments field.

Rules for Coding

1. The system recognizes a delimiter other than /* if you code the DLM parameter on the DD statement defining the data.
2. Code /* (or the value assigned in the DLM parameter) in columns 1 and 2, followed by any comments you have. The comments cannot be continued.
3. The beginning of data to be submitted through an input stream is indicated by a DD * or DD DATA statement.

If the data is preceded by a DD * statement and you do not code the DLM parameter, you do not need to code a delimiter statement.

Example of the Delimiter Statement

```
//JOB54      JOB      , 'C BROWN' ,MSGLEVEL=( 2,0 )
//STEP1     EXEC     PGM=SERS
//DD1       DD      *
            .
            .
            data
            .
            .
/* END OF DATA FOR THIS STEP
```

Section VIII: The Null Statement

Control Statement

The null statement indicates that the job just read should be placed on the queue of jobs ready for processing.

The null statement consists only of the characters // in columns 1 and 2. The remainder of the statement must be blank.

Rules for Coding

1. You can place a null statement at the end of a job's control statements or at the end of all the statements in an input stream.
2. 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.
3. If your job is the last job in the input stream and it is not followed by a null statement, the system recognizes it as the last job in the input stream and places it on the queue.
4. The system flushes any control statements or data between a null statement and the next JOB statement.

Example of the Null Statement

```
//MYJB      JOB      , 'C BROWN'  
//STEP1    EXEC     PROC=FIELD  
//STEP2    EXEC     PGM=XTRA  
//DD1      DD       UNIT=2400  
//DD2      DD       *  
.  
.  
data  
.  
.  
/*  
//
```

Section IX: The PEND Statement

Control Statement

The PEND statement marks the end of an in-stream procedure.

```
//name          PEND      comments
```

The PEND statement consists of the characters // in columns 1 and 2 and three fields: name field, operation (PEND) field, comments.

Rules for Coding

1. Code // in columns 1 and 2, then code a name (1 to 8 characters), or one or more blanks.
2. If you code a name, follow it with one or more blanks.
3. Code PEND and follow it with one or more blanks.
4. Code any desired comments.
5. Do not continue a PEND statement. The PEND statement terminates an in-stream procedure at that point, whether or not the statement is continued.

Examples of the PEND Statement

```
//PROCEND1 PEND THIS STATEMENT IS REQUIRED
```

This PEND statement contains a comment.

```
//          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

Control Statement

The PROC statement is the first control statement in an in-stream procedure. The PROC statement can also be the first control statement in a cataloged procedure. In either an in-stream procedure or a cataloged procedure, a PROC statement can be used to assign default values to symbolic parameters in the procedure.

For additional information on the PROC statement, see *OS/VSI JCL Services*, GC24-5100 and *OS/VSI Planning and Use Guide*, GC24-5090.

//name	PROC	operands	comments
--------	------	----------	----------

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

Rules for Coding

1. A PROC statement is required for an in-stream procedure, and it must appear as the first control statement of the in-stream procedure.

A PROC statement is optional for a cataloged procedure. If you include a PROC statement in a cataloged procedure, it must appear as the first control statement.

2. Code // in columns 1 and 2; then code a 1-to-8-character name, or one or more blanks.
3. If you code a name, follow it with one or more blanks. A name is required for in-stream procedures. Then code PROC, followed by one or more blanks.
4. In the operand field, code the symbolic parameters and their default values. Code a comma after a symbolic parameter and its default value, if you are coding more than one. Do not code a comma after the last symbolic parameter and its default value.
5. Follow the operands with one or more blanks and any desired comments.
6. You can continue the PROC statement onto another statement. Code // in columns 1 and 2 of the continuation statement.
7. To assign a value to a symbolic parameter, code:

```
symbolic parameter=value
```

Omit the ampersand that precedes the symbolic parameter in the procedure.

8. The value you assign to a symbolic parameter can be any length, but it cannot be continued onto another statement.
9. If the symbolic parameter value contains special characters, enclose the value in apostrophes (they will not be considered part of the value).

If the special characters include apostrophes, you must code each apostrophe as two consecutive apostrophes.

10. If you assign more than one value to a symbolic parameter on the PROC statement, the first value encountered is assigned.
11. If you concatenate the symbolic parameter with some other information, (for example, `&JOBNO.321`), the information and value cannot exceed a total of 120 characters.
12. You can override a default value appearing on a PROC statement by assigning a value to the same symbolic parameter on the EXEC statement that calls the procedure.

Examples of the PROC Statement

```
//DEF      PROC      STATUS=OLD, LIBRARY=SYSLIB,  
//          NUMBER=777777  
//NOTIFY   EXEC      PGM=ACCUM  
//DD1      DD        DSNAME=MGMT, DISP=( &STATUS, KEEP ),  
//          UNIT=2400, VOLUME=SER=888888  
//DD2      DD        DSNAME=&LIBRARY, DISP=( OLD, KEEP ),  
//          UNIT=2314, 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 you have not assigned values to the symbolic parameters.

```
//CARDS    PROC
```

This PROC statement can be used to mark the beginning of an in-stream procedure named CARDS.

Appendix A: Identifying Data Sets to the System

Specifying the DDNAME Parameter

The DDNAME parameter is most often used in cataloged procedures and in job steps that call procedures.

It is used in cataloged 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, that is, 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
```

it saves XYZ and, temporarily, PHOB. The data set is a dummy data set, until the ddname is encountered in the input stream.

When the system encounters a statement whose ddname has been temporarily saved, it does two things: (1) it uses the information contained on this statement to define the data set and (2) 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=2314,
//          VOLUME=SER=46231,SPACE=(TRK,(20,5))
          .
          .
//DD12    DD      DSN=SET13,DISP=(NEW,KEEP),
//          VOLUME=REF=*.DD1,
//          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=2314,
//          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.

DCB Subparameters BLKSIZE and BUFNO

Two DCB subparameters that can be coded with the DDNAME parameter are BLKSIZE and BUFNO. You can assign these DCB characteristics to the data set defined in the referenced DD statement. When the DCB subparameters BLKSIZE and BUFNO are coded on both the DD statement that contains the DDNAME parameter and on the referenced DD statement, the subparameters coded on the former are ignored.

These subparameters are most often coded with the DDNAME parameter when the referenced DD statement defines data in the input stream. Data in the input stream is written on a direct access device, and the records are blocked as they are written. The input reader procedure normally assigns a blocksize 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.)

Specifying the DSNAME Parameter

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 permanent 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 Sets

When a nontemporary data set is created, it is assigned a name in the DSNAME parameter and 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 one through eight 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.

A qualified data set name consists of one 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 one through 35 characters (including periods). For each eight characters or fewer, 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.

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 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, use the IEHPROGM utility program to delete or revise the pointer in the control volume after the job has terminated. 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 *OS/VS Utilities*, GC35-0005.

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, specify the partitioned data set name and follow it with the member name. The member name is enclosed in parentheses and consists of one to eight characters. The first character must be an alphabetic or national (@, \$, #) character; the remaining characters can be any alphameric or national characters.

A Generation Data Group

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, specify the generation data group name and follow it with the generation number. The generation number is enclosed in parentheses and is a zero or a signed integer. A zero represents the most current generation of the group; a negative integer (for example, -1) represents an older generation; a positive integer (for example, +1) represents a new generation that has not yet been cataloged.

To retrieve all generations of a generation data group (up to 255 generations), code only the group name in the DSNNAME parameter and the DISP parameter.

A complete discussion of creating and retrieving generation data sets is contained in *OS/VS1 JCL Services*, GC24-5100.

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 DSNNAME=dsname or DSNNAME=dsname(PRIME). When you retrieve the data set, code only the data set name; do not include the term PRIME, INDEX, or OVFLOW.

Creating or Retrieving a Temporary Data Set

If the data set is temporary, you can identify:

- A temporary data set by coding `DSNAME=εεdsname`.
- A member of a temporary partitioned data set by coding `DSNAME=εεdsname(member name)`.
- An area of a temporary indexed sequential data set by coding `DSNAME=εεdsname(area name)`.

Note: You may make a backward reference to a *SYSIN* or *SYSOUT* data set by using a *DSNAME* reference.

Temporary Data Sets

A temporary data set is any data set that is created and deleted within the same job. 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 one through eight characters and is preceded by two ampersands (εε). 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.)

The system generates a qualified name for the temporary data set, consisting of:

1. 'SYS' followed by the Julian date
2. 'T' followed by a time stamp
3. two identifying characters followed by a three-byte protection key
4. the job name
5. the name portion of εεname, if specified. Otherwise, an eight-byte name formed from an identifying character and a seven-byte number that increases for each temporary data set within a job.

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.

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, 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 one through eight 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=εεdsname` or `DSNAME=εεdsname(PRIME)`. If you want to retrieve the temporary data set on the same job, code only the data set's temporary name; do not include the term PRIME, INDEX, or OVFLOW.

Using a Dedicated Data Set

If your installation provides dedicated data sets, you can use them 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, because the data sets are already allocated.

To use a dedicated data set, code `DSNAME=εεname` or `DSNAME=εname` on a DD statement, along with all other parameters required to define your temporary data set, for example, 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 this dedicated data set, the parameters coded on your DD statement are used to create a temporary data set.

Identifying Associated Data Sets

Associated data sets are data sets residing on diskette that are separate from the instream data and are to be spooled as SYSIN data sets. Associated data sets are identified by specifying a data set identifier (DSID) and, optionally, volume identifier on the DD * or DD DATA statement in the job stream.

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 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, code `DSNAME=*.stepname.ddname`. When the earlier DD statement is contained in the same job step, code `DSNAME=*.ddname`. When the earlier DD statement is contained in a cataloged procedure step called by an earlier job step, code `DSNAME=*.stepname.procstepname.ddname`.

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; for example, `DSNAME='DAT+5'`. If one of the special characters is an apostrophe, you

must identify it by coding two consecutive apostrophes in its place; for example, DSNNAME='DAYS"END'. A data set name enclosed in apostrophes can consist of one through 44 characters.

There are cases when your data set name must contain required special characters, that tell the system something about the data set (for example, $\epsilon\epsilon$ in DSNNAME= $\epsilon\epsilon$ 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, *must not* be enclosed in apostrophes:

- DSNNAME=name(member name)
- DSNNAME=name(area name)
- DSNNAME=name(generation number)
- DSNNAME= $\epsilon\epsilon$ name
- DSNNAME=*.stepname.ddname

Keep these 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 system assigns a temporary data set name.
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 do not need to enclose the data set name in apostrophes.

Specifying the LABEL Parameter

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 user 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.

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`).

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, that is, 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 one-to-four-digit number. The system assumes 1, (the first data set on the reel) if you omit this subparameter. If you code 0 and the data set is not a passed or cataloged data set, 1 is also assumed. 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. See *OS/VS Tape Labels*, GC26-3795, for further information on tapemark positioning.

Label Type Subparameter

The label type subparameter tells the system what label type 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.
- `AUL` – if the data set has American National Standard labels and American National Standard user labels.
- `NSL` – if the data set has nonstandard labels.

- NL – if the data set has no labels.
- BLP – if you want label processing bypassed.
- LTM – leading tape mark (OS/DOS interchange).

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 or BLP, the operator must ensure that the correct tape volume is mounted. If you specify NL, the data set must have no standard labels. When you specify AL or AUL, the system ensures that the correct American National Standard labeled tape is mounted.

For cataloged and passed data sets, label type information is not kept. Therefore, any time you refer to a cataloged or passed 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.

Note for BLP: When you request the system to bypass label processing and the tape volume has labels, the system treats anything between tape-marks 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. See *OS/VS Tape Labels*, GC26-3795, for further information on tapemark positioning.

The label type subparameter can also be specified when you make a nonspecific volume request for a tape volume (that is, no volume serial numbers are specified on the DD statement), and you want the data set to have a certain type of label. 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; and (3) you make a nonspecific volume request. If all these conditions are not met, 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.

PASSWORD and NOPWREAD Subparameter

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 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.

IN and OUT Subparameters

BSAM (basic sequential access method) 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, that is, 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.

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 a data set, you can use the IEHPROGM utility program to delete the data set. The IEHPROGM utility program is described in *OS/VS Utilities*, GC35-0005.)

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 subparame-

ter, or (2) tell the system the exact date after which the data set need not be kept, the EXPDT subparameter.

If you code the RETPD subparameter, you specify a one-to-four-digit number, which represents the number of days the data set is to be kept. If you code the EXPDT subparameter, specify a two-digit year number and a three-digit day number (for example, January 1 would be 001, July 1 would be 182), which represents the date after which the data set need not 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.

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.

Appendix B: Reference Tables

Figures 4, 5, and 6 summarize the DD statement parameters required to perform these functions:

- Create a data set on a unit record device (card punch or printer)
- Create a data set on a system output device
- Create a data set on magnetic tape
- Create a data set on a direct access device
- Retrieve a data set from a unit record device (card reader or paper tape reader)
- Retrieve a data set from the input stream
- Retrieve a passed data set (magnetic tape or direct access)
- Retrieve a cataloged data set (magnetic tape or direct access)
- Retrieve a kept data set (magnetic tape or direct access)
- Extend a passed data set (magnetic tape or direct access)
- Extend a cataloged data set (magnetic tape or direct access)
- Extend a kept data set (magnetic tape or direct access)

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 Options	OUTLIM	Optional. Meaningful only for systems that utilize the Systems Management Facilities option.
		COPIES	Optional
		DSID	Required for output to a 3540.
		HOLD	Optional
DEST		Optional	
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 will 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	
		CHKPT	Optional
		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	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	
		CHKPT	Optional
		DUMMY	Optional

Figure 4. DD Parameters for Creating a Data Set (Part 1 of 2)

Device	Parameter Type	Parameter	Comments
Mass Storage System	Data Set Information	DSNAME (or DSN)	Required if the data set is to be cataloged or used by another 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 Data Set	UNIT	Required, must be 3330V.
		VOLUME	VOL=SER is mutually exclusive with MSVGP. If you specify VOL=SER, SPACE is required. VOL=PRIVATE guarantees allocation to SYSGROUP.
		LABEL	Required if you want the data set to have both IBM standard and user labels.
		MSVGP	Required if you want allocation to SYSGROUP. Specify MSVGP=SYSGROUP. Mutually exclusive with SER,SYSOUT,COPIES, DDNAME,ABSTR, and DSID.
	Size of Data Set	SPACE	Required if you code neither MSVGP nor VOL=SER. Also required when VOL=SER is coded. If you code MSVGP and do not code SPACE, the default is CONTIG. You must code SPACE to get noncontiguous space allocation when MSVGP is coded.
	Data Attributes	DUMMY	Optional

Figure 4. DD Parameters for Creating a Data Set (Part 2 of 2)

Data Set	Parameter Type	Parameter	Comments
Unit Record Devices	Location of the Data Set	UNIT	Required
	Data Attributes	DCB	Optional
	Special Processing Option	DUMMY	Optional
Input Stream	Location of the Data Set	*	You must specify <u>one</u> of these parameters.
		DATA	
	Data Attributes	DCB	Optional
	Special Process Option	DLM	Optional
Associated Data Set	Location of the Data Set	*	You must specify <u>one</u> of these parameters.
		DATA	
	Data Set Information	DSID	Required for 3540 associated data sets.
		VOL=SER	Optional for 3540 associated data sets.
	DCB=LRECL	Optional for 3540 associated data sets.	
Passed Data Set	Data Set Information	DSNAME	Required
		DISP	Required
	Location of the Data Set	UNIT	Required only if you want more units.
		LABEL	Required only if the data set does not have IBM standard labels.
	Data Attributes	DCB	Optional
	Special Processing Options	CHKPT	Optional
DUMMY		Optional	
Cataloged Data Set	Data Set Information	DSNAME	Required
		DISP	Required
	Location of the Data Set	UNIT	Optional
		VOLUME	May be required if you want to begin processing with a volume after the first.
		LABEL	Required only if the data set does not have IBM standard labels.
	Data Attributes	DCB	Optional
	Special Processing Options	SEP	Either parameter can be used.
		AFF	
		CHKPT	Optional
		DUMMY	Optional
Kept Data Set	Data Set Information	DSNAME	Required
		DISP	Required
	Location of the Data Set	UNIT	Required
		VOLUME	Required
		LABEL	Required only if the data set does not have IBM standard labels.
	Data Attributes	DCB	Optional
	Special Processing Options	SEP	Either parameter can be used.
		AFF	
CHKPT		Optional	
DUMMY		Optional	

Figure 5. DD Parameters for Retrieving a Data Set

Data Set	Parameter Type	Parameter	Comments
Passed Data Set	Data Set Information	DSNAME	Required
		DISP	Required
	Location of the Data Set	UNIT	Required only if you want more units.
		VOLUME	Required only if you need more volumes.
		LABEL	Required only if the data set does not have IBM standard labels.
	Size of the Data Set	SPACE	Required only if you want to override the secondary quantity.
	Data Attributes	DCB	May be required if data set does not have IBM standard labels.
	Special Processing Options	CHKPT	Optional
DUMMY		Optional	
Cataloged Data Set	Data Set Information	DSNAME	Required
		DISP	Required
	Location of the Data Set	UNIT	Optional
		VOLUME	Required only if you need more volumes.
		LABEL	Required only if the data set does not have IBM standard labels.
	Size of the Data Set	SPACE	Required only if you want to override the secondary quantity.
	Data Attributes	DCB	Required only if the data set does not have IBM standard labels.
	Special Processing Options	SEP	Either parameter can be used.
		AFF	
		CHKPT	Optional
DUMMY		Optional	
Kept Data Set	Data Set Information	DSNAME	Required
		DISP	Required
	Location of the Data Set	UNIT	Required
		VOLUME	Required
		LABEL	Required only if data set does not have IBM standard labels.
	Size of the Data Set	SPACE	Required only if you want to override the secondary quantity.
	Data Attributes	DCB	Required only if the data set does not have IBM standard labels.
	Special Processing Options	SEP	Either parameter can be used.
AFF			
CHKPT		Optional	
DUMMY		Optional	

Figure 6. DD Parameters for Extending a Data Set

Retrieving or Extending an Indexed Sequential Data Set

A maximum of three DD statements are needed to retrieve or extend an ISAM (indexed sequential access method) data set. The DD statements are coded in this order:

1. First DD statement – defines the indexed area
2. Second DD statement – defines the prime area
3. Third DD statement – defines the overflow area

Only the second DD statement is required. The first DD statement is not needed if either the index area resides on a volume of the same type as the prime area or if the index area is part of the prime area. The third DD statement is not needed if either the overflow area resides on a volume of the same type as the prime area or if there is no overflow area. When the first or third DD statements, or both, are not needed, the definition of the index or overflow areas are included in the DD statement for the prime area.

Area	Parameter Type	Parameter	Comments
Index (used only if index area not on same device type as prime area) (First DD statement)	Data Set Information	DSNAME	Required. You must code the same value as in second DD statement.
		DISP	Required. You must code the same value as in second DD statement.
	Location of the data set	UNIT	Required
		VOLUME	Required
	Data Attributes	DCB	Required
Prime and Overflow; or Index, Prime, and Overflow; or Index and Prime (required) (Second DD statement)	Data Set Information	DSNAME	Required
		DISP	Required. Specifies whether you are retrieving the data set.
	Location of the data set	UNIT	Required unless it is a passed data set with all three areas on one volume.
		VOLUME	Same requirement as UNIT. If used, code volumes in order they were defined.
	Data Attributes	DCB	Required
Overflow (used only if overflow area not on same device type as prime area) (Third DD Statement)	Data Set Information	DSNAME	Required. You must code the same value as in second DD statement.
		DISP	Required. You must code the same value as in the second DD statement.
	Location of the data set	UNIT	Required
		VOLUME	Required
	Data Attributes	DCB	Required

Figure 7. DD Parameters for Retrieving or Extending an Indexed Sequential Data Set

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 8. Area Arrangement of Indexed Sequential Data Sets

	AFF	AMP ¹	BURST	CHARS	CHKPT	COMPACT	COPIES	DATA	DCB	DDNAME	DEST	DISP	DLM	DSID	DSN	DSNAME	DUMMY	FCB	FLASH	HOLD	LABEL	MODIFY	MSVGP	OUTLIM	QNAME	SEP	SPACE	SPLIT	SUBALLOC	SUBSYS	SYSOUT	TERM	UCS	UNIT	VOL	VOLUME	*		
AFF	•						•	•		•									•			•				•											•		
AMP ¹		•	•	•			•		•									•	•			•				•	•												
BURST	•	•	•					•		•		•		•	•	•						•		•		•										•	•	•	
CHARS	•	•		•				•		•		•		•	•	•						•		•		•										•	•	•	
CHKPT					•			•		•																				•	•							•	
COMPACT						•		•		•																												•	
COPIES	•	•					•	•		•		•		•	•	•						•		•		•										•	•	•	
DATA	•		•	•	•	•	•	•		•		•			•	•	•	•	•	•	•	•		•		•	•	•	•	•	•	•	•	•	•	•	•	•	
DCB		•							•																													•	
DDNAME	•		•	•	•	•	•	•		•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	
DEST								•		•	•																											•	
DISP			•	•			•	•		•		•										•																•	
DLM													•																										
DSID			•	•			•							•						•			•	•															
DSN			•	•			•	•		•					•	•						•		•		•												•	
DSNAME			•	•			•	•		•					•	•						•		•		•												•	
DUMMY							•		•								•																					•	
FCB		•					•		•									•																				•	
FLASH	•	•					•		•		•		•	•	•					•		•		•		•	•	•	•	•	•	•	•	•	•	•	•	•	
HOLD								•		•											•																	•	
LABEL			•	•			•	•		•										•		•	•															•	
MODIFY	•	•					•		•		•		•	•	•							•	•	•		•	•	•	•	•	•	•	•	•	•	•	•	•	
MSVGP			•	•			•		•					•						•			•	•														•	
OUTLIM								•		•														•														•	
QNAME		•	•	•			•	•		•					•	•						•				•												•	
SEP	•		•	•			•	•		•										•		•					•											•	
SPACE		•						•		•																		•	•	•	•	•	•	•	•	•	•	•	•
SPLIT		•	•	•			•	•		•										•		•						•	•	•	•	•	•	•	•	•	•	•	
SUBALLOC		•	•	•			•	•		•										•		•	•					•	•	•	•	•	•	•	•	•	•	•	
SUBSYS	•	•	•	•	•	•	•	•		•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	
SYSOUT	•	•			•		•		•		•											•		•		•	•	•	•	•	•	•	•	•	•	•	•	•	
TERM		•					•		•																													•	
UCS		•						•		•																												•	
UNIT								•		•																												•	
VOL			•	•			•			•										•			•														•	•	
VOLUME			•	•			•			•										•			•														•	•	
*	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	

LEGEND:

The horizontal and the vertical DD parameters are mutually exclusive, that is, they cannot be coded together on one DD statement.

As indicated by the table, each DD parameter is mutually exclusive with itself, that is, it cannot appear twice on the same DD statement.

¹ Additional information can be found in Appendix D of *OS/VS Virtual Storage Access Method (VSAM) Programmer's Guide GC26-3838*.

Make SUBSYS mutually exclusive with all parameters except DCB.

Figure 9. Table of Mutually Exclusive DD Parameters

Figure 10. Disposition Processing Chart

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
				A ²	B ³	C ⁴	
NEW or MOD ⁵	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 ⁶	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 In the following cases, the data set is not allocated to the job step and, therefore, no disposition processing is performed: a JCL error is encountered; a return code test causes the job step to be bypassed; the job is canceled before data set allocation; the system cannot allocate this data set to the job step.
- 3 This is the disposition processing that is performed when the job is canceled after data set allocation or a processing program error occurs.
- 4 This is the disposition processing that is performed when this data set has been allocated to the step but the system cannot allocate some other data set to the job step.
- 5 For MOD, a data set is considered to be a new data set if volume information is not available to the system.
- 6 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 non-specific request for a tape volume with a 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.

Device	2314 (each volume)	2305	3330 *	3330-1 *	3340	3350
Storage Medium	Disk	Disk	Disk	Disk	Disk	Disk
Cylinders	200	Model 1: 48 Model 2: 96	404	808	696 (70-megabytes) 348 (35-megabytes)	555
Tracks Per Cylinder	20	8	19	19	12	30
Bytes Per Track	7,294	Model 1: 14,136 Model 2: 14,660	13,030	13,030	8,368	19,069
Bytes Per Cylinder	145,880	Model 1: 113,088 Model 2: 117,280	247,570	247,570	100,416	572,070
Bytes Per Device (in millions)	29.17	Model 1: 5.4 Model 2: 11.3	101.6	201.7	69.8 (70-megabytes) 34.9 (35-megabytes)	317.5

* 3330 is the device type for 3330 Model 1 and 3333 Model 1. 3330-1 is the device type for 3330 Model 11 and 3333 Model 11.

Figure 11. Direct Access Capacities

Maximum Bytes per Record Formatted without Keys						Records per Track	Maximum Bytes per Record Formatted with Keys					
2314	2305-1	2305-2	3330/ 3330-1 *	3340	3350		2314	2305-1	2305-2	3330/ 3330-1 *	3340	3350
7294	14136	14660	13030	8368	19069	1	7249	13934	14569	12974	8293	18987
3520	6852	7231	6447	4100	9442	2	3476	6650	7140	6391	4025	9360
2298	4424	4754	4253	2678	6233	3	2254	4222	4663	4197	2603	6151
1693	3210	3516	3156	1966	4629	4	1649	3008	3425	3100	1891	4547
1332	2480	2773	2498	1540	3665	5	1288	2278	2682	2442	1465	3583
1092	1996	2278	2059	1255	3024	6	1049	1794	2187	2003	1180	2942
921	1648	1924	1745	1052	2565	7	877	1446	1833	1689	977	2484
793	1388	1659	1510	899	2222	8	750	1186	1568	1454	824	2140
694	1186	1452	1327	781	1955	9	650	984	1361	1271	706	1873
615	1024	1287	1181	686	1740	10	571	822	1196	1125	611	1658
550	892	1152	1061	608	1565	11	506	690	1061	1005	533	1483
496	782	1040	962	544	1420	12	452	580	949	906	469	1338
450	688	944	877	489	1296	13	407	486	853	821	414	1214
411	608	863	805	442	1190	14	368	406	772	749	367	1108
377	538	792	742	402	1099	15	333	336	701	686	327	1017
347	478	730	687	366	1018	16	304	276	639	631	291	936
321	424	676	639	335	948	17	277	222	585	583	260	866
298	376	627	596	307	885	18	254	174	536	540	232	803
276	334	584	557	282	828	19	233	132	493	501	207	746
258	296	544	523	259	778	20	215	94	453	467	184	696
241	260	509	491	239	732	21	198	58	418	435	164	650
226	230	477	463	220	690	22	183		386	407	145	608
211	200	448	437	204	652	23	168		357	381	129	570
199	174	421	413	188	617	24	156		330	357	113	535
187	150	396	391	174	585	25	144		305	335	99	503
176	128	373	371	161	556	26	133		282	315	86	474
166	106	352	352	149	528	27	123		261	296	74	446
157	88	332	335	137	503	28	114		241	279	62	421
148	70	314	318	127	479	29	105		223	262	52	397
139	52	297	303	117	457	30	96		206	247	42	375

* 3330 is the device type for 3330 Model 1 and 3333 Model 1. 3330-1 is the device type for 3330 Model 11 and 3333 Model 11.

Figure 12. Track Capacities

	Character Arrangement Table Names	Character Set	Pitch 1	Number of Graphic Characters 2	Comments
Basic Group	GS10	Gothic-10	10	63	
	GS12	Gothic-12	12	63	
	GS15	Gothic-15	15	63	
	GSC	Gothic-15 condensed	15	63	
	GF10	Gothic-10 folded	10	62	3
	GF12	Gothic-12 folded	12	62	3
	GF15	Gothic-15 folded	15	62	3
	GFC	Gothic-15 condensed & folded	15	62	3
	GU10	Gothic-10 underscored	10	63	
	GU12	Gothic-12 underscored	12	63	
	GU15	Gothic-15 underscored	15	63	
	GUC	Gothic-15 condensed & underscored	15	63	
TU10	Text 1 & 2 underscored	10	120	Uses two WCGMs 4	
DUMP	Gothic-15 & underscored Gothic-15	15	79	Uses two WCGMs 4	
3211 Group	A11	Gothic-10	10	48	
	G11	Gothic-10	10	63	
	H11	Gothic-10	10	48	
	P11	Gothic-10	10	60	
	T11	Text 1 & 2	10	120	Uses two WCGMs 4
1403 Group	AN	Gothic-10	10	48	
	GN	Gothic-10	10	63	
	HN	Gothic-10	10	48	
	PCAN	Gothic-10	10	48	
	PCHN	Gothic-10	10	48	
	PN	Gothic-10	10	60	
	QN	Gothic-10	10	60	
	QNC	Gothic-10	10	60	
	RN	Gothic-10	10	52	
	XN	Gothic-10	10	40	
	YN	Gothic-10	10	42	
	SN	Text 1 & 2	10	84	Uses two WCGMs 4
	TN	Text 1 & 2	10	120	Uses two WCGMs
OCR Group	AOA	OCR-A, Gothic-10	10	48	Uses two WCGMs
	AOD	OCR-A, Gothic-10	10	48	Uses two WCGMs
	AON	OCR-A, Gothic-10	10	48	Uses two WCGMs
	OAA	OCR-A, Gothic-10	10	48	Uses two WCGMs
	ODA	OCR-A, Gothic-10	10	48	Uses two WCGMs
	ONA	OCR-A, Gothic-10	10	48	Uses two WCGMs
	BOA	OCR-B, Gothic-10	10	48	Uses two WCGMs
	BON	OCR-B, Gothic-10	10	48	Uses two WCGMs
	OAB	OCR-B	10	48	
	ONB	OCR-B, Gothic-10	10	48	Uses two WCGMs
Katakana Group	2773	Katakana-10, Gothic-10	10	62	Uses two WCGMs
	2774	Katakana-10, Gothic-10	10	108	Uses two WCGMs
	KN1	Katakana-10, Gothic-10	10	127	Uses two WCGMs
Format Group	FM10	Format-10	10	36	
	FM12	Format-12	12	36	
	FM15	Format-15	15	36	

1 For any table using 10-pitch Gothic or Katakana, the pitch can be changed to 12 or 15 by changing the character set identifier using the IEBIMAGE utility. (For the GN and G11 tables, a 12-pitch or 15-pitch graphic character modification module equivalent to SPC1 is also required.)

2 Does not include a blank, which is also part of each character set except Katakana.

3 The GF10, GF12, GF15, and GFC tables provide the folding effect to allow the printing of uppercase graphic characters when lowercase are called for.

4 A WCGM (writable character generation module) is a 64-position portion of character generation storage for one character set.

Figure 13. Character Arrangement Tables Supplied with the 3800

Appendix C: Device Types

The programmer specifies each I/O device in the UNIT parameter of a sysgen IODEVICE macro instruction. The system assigns each device (except telecommunication devices) to a device type. Each device type includes one or more devices that have a distinctive set of features. The device type generated is for use in the UNIT parameter of the DD statement. You can code only those device types that were defined during system generation.

The following table lists the device types that can be coded in the UNIT parameter.

Device Type	Description
Magnetic Tape Devices	
2400	Any 9-track magnetic tape unit having an 800 bits-per-inch (density) capability when the dual-density feature is not installed or a 1600 bits-per-inch (density) capability when the dual-density feature is installed
2400-1	2400 series magnetic tape unit with 7-track capability and without data conversion
2400-2	2400 series magnetic tape unit with 7-track capability and data conversion
2400-3	2400 series or 2420 9-track magnetic tape unit having only a 1600 bits-per-inch (density) capability
2400-4	2400 series 9-track magnetic tape drive having an 800 and 1600 bits-per-inch (density) capability
2495	2495 Tape Cartridge Reader
3400-2	3420 Magnetic Tape Unit having 7-track capability and data conversion
3400-3	3410 or 3420 9-track Magnetic Tape Unit having 1600 bits-per-inch (density) capability
3400-4	3410 or 3420 9-track Magnetic Tape Unit having 800 and 1600 bits-per-inch (density) capability
3400-5	3420 9-track Magnetic Tape Unit having 6250 bits-per-inch (density) capability
3400-6	3420 9-track Magnetic Tape Unit having 1600 and 6250 bits-per-inch (density) capability
Direct-Access Devices	
2305-1	2305 Fixed Head Storage Model 1
2305-2	2305 Fixed Head Storage Model 2
2314	2314/2319 Disk Storage
3330	3330 Disk Storage Model 1 and 3333 Disk Storage and Control Model 1

3330-1 3330 Disk Storage Model 11 and 3333 Disk Storage and Control Model 11
3330V 3330 Disk Storage Model 1 or 11 and the 3333 Disk Storage and Control Model 1 or 11 when used for MSS
3340 3340 Direct Access Storage Facility
3350 3350 Direct Access Storage

Unit Record Equipment and Other Devices

1052 1052 Printer Keyboard Model 7
1053 1053 Printer Model 4
1403 1403 Printer
1442 1442 Card Read Punch
1443 1443 Printer Model N1
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
3036 3036 Display Console Keyboard - 3031, 3032, and 3033 Processors
3066 3066 System Console - System/370 Model 165II and 168
3138 3138 Display Console Keyboard - System/370 Model 138
3148 3148 Display Console Keyboard - System/370 Model 148
3158 3158 Display Console Keyboard - System/370 Model 158
3203-4 3203 Printer
3210 3210 Console Printer Keyboard
3211 3211 Printer
3213 3213 Console Printer - System/370 Model 155II
3215 3215 Console Printer Keyboard
3278-2A 3278-2A Display Console Keyboard - 4331 and 4341 Processors
3505 3505 Card Reader
3525 3525 Card Punch
3540 3540 Diskette I/O Unit
3800 3800 Printing Subsystem
3851 3851 Mass Storage Facility
5098-5 5098 Sense Base Control Unit
7443 7443 Service Record File Model 1

Graphics Devices

2250-1	2250 Display Unit Model 1
2250-3	2250 Display Unit Model 3
2260-1	2260 Display Station (local attachment) Model 1
2260-2	2260 Display Station (local attachment) Model 2
2265	2265 Display Station
3277-1	3277 Display Station Model 1
3277-2	3277 Display Station Model 2
3284-1	3284 Printer Model 1
3284-2	3284 Printer Model 2
3286-1	3286 Printer Model 1
3286-2	3286 Printer Model 2

Optical Character Readers

1275	1275 Optical Reader Sorter (available through World Trade branch offices only)
1287	1287 Optical Reader
1288	1288 Optical Page Reader
3886	3886 Optical Character Reader

Reader/Inscriber

3895	3895 Document Reader/Inscriber
------	--------------------------------

Magnetic Character Reader

1419	1419 Magnetic Character Reader
3890	3890 Document Processor

Audio Response

7770	7770 Audio Response Unit
------	--------------------------

Remote Analysis

2955	2955 Remote Analysis Unit
------	---------------------------

Communications Controllers

3704	3704 Communications Controller
3705	3705 Communications Controller
3791L	3971 Controller

Format Charts

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
		[ADDRSPC={ VIRT { REAL } }	K	Requests storage type
		[CLASS=jobclass]	K	Assigns A-Z, 0-9
		[COND=((code, operator), ...)]	K	Specifies a maximum of 8 tests
		[MPROFILE='profile string']	K	For ISSP only
		[MSGCLASS=output class]	K	Assigns A-Z, 0-9
		[MSGLEVEL={ 0 [, 0] 1 [, 1] 2 }]	K	
		[PROFILE='profile string']	K	For ISSP only
		[PRTY=priority]	K	Assigns 0-13
		[RD={ R RNC NC NR }]	K	Restart definition
		[REGION=valueK]	K	Specifies amount of storage space
		[RESTART={ * stepname stepname.procstepname } [, checkid]]	K	For deferred restart
		[TIME= { (minutes)[,seconds] } 1440 }]	K	Assigns job CPU time limit
		[TYPRUN={ HOLD { SCAN } }]	K	Holds a job in job queue, or scans JCL for syntax errors
Legend : P Positional parameter. K Keyword parameter. { } Choose one. [] Optional; if more than one line is enclosed, choose one or none.				

Figure 14. The JOB Statement

The EXEC Statement				
//Name	Operation	Operand	P/K	Comments
//[stepname]	EXEC	<pre> PGM= { program name { *.stepname.ddname { *.stepname.procstepname.ddname } } [PROC=]procedure name [ACCT=(accounting information,...) [ACCT.procstepname=(accounting information,...)] [ADDRSPC={ VIRT { REAL { } [COND={ (code,operator) { (code,operator,stepname) { (code,operator,stepname.procstepname) } ,...[, [EVEN] { (code,operator,stepname.procstepname) } ONLY]) [COND.procstepname={ (code,operator) { (code,operator,stepname) { (code,operator,stepname.procstepname) } ,...[, [EVEN] { (code,operator,stepname.procstepname) } ONLY]) [PARM=value [PARM.procstepname=value [RD= { R { RNC { NC { NR [RD.procstepname={ R { RNC { NC { NR [REGION=valueK [TIME= { {[minutes][,seconds]} { 1440 [TIME.procstepname= { (minutes,seconds) { 1440 </pre>	<p>P Identifies program or cataloged procedure</p> <p>K Accounting information for step</p> <p>K Requests storage type</p> <p>K Specifies a maximum of 8 tests, or 7 tests if EVEN or ONLY is coded</p> <p>K Parentheses or apostrophes enclosing value may be required</p> <p>K Restart definition</p> <p>K Specifies amount of storage space</p> <p>K Assigns step CPU time limit</p>	
Legend: P Positional parameter. K Keyword parameter. {} Choose one. [] Optional; if more than one line is enclosed, choose one or none.				

Figure 15. The EXEC Statement

The DD Statement

//Name	Operation	Operand	P/K	Comments
<pre>//ddname procstepname ddname</pre>	DD	<pre>* DATA[,DLM=xx] [DUMMY] [AFF=ddname] AMP={ 'AMORG' [, 'BUFND=number'] [, 'BUFNI=number'] [, 'BUFSP=number'] [, 'CROPS= { 'NCK' 'NRC' 'NRE' 'RCK' }] [, 'OPTCD= { 'I' 'L' 'IL' }] [, 'RECFM= { 'F' 'FB' 'V' 'VB' }] [, 'STRNO=number'] [, 'SYNAD=modulename'] [, 'TRACE'] } [BURST= { Y N }]¹ [CHARS=(table name, ...)]¹ [CHKPT=EOV] [COMPACT= { NO compact table id }] [COPIES=(nnn[, (group value, ...)])] [DCB=(list of attributes) DCB={ dsname * .ddname * .stepname.ddname * .stepname.procstepname.ddname } [,list of attributes)] [DDNAME=ddname] [DEST=userid] DISP={ [NEW [,DELETE] [OLD [,KEEP] [SHR [,PASS] [MOD [,CATLG] [,UNCATLG] [,UNCATLG] }] [DLM=delimiter] [DSID=(id[,V])¹ { 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 } [FCB=(image-id [,ALIGN ,VERIFY])¹ [FLASH=(overlay name [,count])]¹ [HOLD={ YES NO }] [LABEL=([data set seq #] [,SL [,SUL [,AL [,AUL [,NSL [,NL [,BLP [,LTM [,PASSWORD] [,NOPWREAD] [,IN [,OUT] [,EXPDT=yyddd [,RETPD=nnnn]])</pre>	<p>P</p> <p>P</p> <p>K</p> <p>K</p> <p>K</p> <p>K</p> <p>K</p> <p>K</p> <p>K</p> <p>K</p> <p>K</p> <p>K</p> <p>K</p> <p>K</p> <p>K</p> <p>K</p> <p>K</p> <p>K</p> <p>K</p> <p>K</p> <p>K</p>	<p>Defines data set in the input stream.</p> <p>Bypasses I/O operations on a data set (BSAM and QSAM).</p> <p>Requests channel separation.</p> <p>Modifies the program processing VSAM clusters or components.</p> <p>Describes printed output.</p> <p>Describes character arrangements used in printing.</p> <p>For checkpoint at EOV</p> <p>Identifies Compaction Table for RES output.</p> <p>For use with the SYSOUT and UNIT parameter.</p> <p>Completes the data control block.</p> <p>Postpones the definition of a data set.</p> <p>Specifies remote destination for SYSOUT data set.</p> <p>Assigns a status, disposition, and conditional disposition to the data set. CATLG, NEW, and UNCATLG are invalid for VSAM components and clusters.</p> <p>Assigns delimiter other than /*.</p> <p>Indicates that there is associated data for this DD statement.</p> <p>Assigns 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. Area generation, member, and temporary names are invalid or VSAM clusters and components.</p> <p>Specifies forms control information. The FCB parameter is ignored if the data set is not written to a 3211 printer.</p> <p>Specifies forms overlay option and copies to be flashed.</p> <p>Specifies whether JES writer processing of a SYSOUT data set is to be deferred or processed normally.</p> <p>Supplies label information.</p>

Legend:
P Positional parameter.
K Keyword parameter.
1 Valid for 3800 only.

Figure 16. The DD Statement (Part 1 of 2)

Figure 16. The DD Statement (Part 2 of 2)

The DD Statement (con't)				
//Name	Operation	Operand	P/K	Comments
// [ddname procstepname. ddname]	DD	<pre> [MODIFY=(module name[,table reference character])] ¹ [MSVGP=(id[,ddname])] [OUTLIM=number] [QNAME=process name] [SEP=(ddname, . . .)] { SPACE=({ TRK CYL blocklength } , (primary [,secondary] [,directory] [,index]) [,RLSE [,CONTIG MXIG ALX] [,ROUND]) } { SPACE=(ABSTR,(primary quantity,address [,directory] [,index])) } ²SPLIT=({ (n,CYL,(primary quantity [,secondary quantity])) n (percent,blocklength,(primary quantity [,secondary quantity])) percent }) ²SUBALLOC=({ TRK CYL blocklength } , (primary [,secondary] [,directory]) { ,ddname stepname.ddname stepname.procstepname.ddname }) [SUBSYS=(name [,parm] ...)] [SYSOUT=({ classname [,program name] [,form number] [,PROFILE='sysout profile string']) } PROFILE='sysout profile string')] [TERM=RT] ²UCS=(character set code [,FOLD] [,VERIFY]) { UNIT=([unit address device type group name] [,unit count P] [,DEFER] [,SEP=(ddname, . . .)]) } [UNIT=AFF=ddname] { VOLUME } = ([PRIVATE] [,RETAIN] [,volume seq #] [,volume count] [,SER=(serial number, . . .) REF=dsname REF=* .ddname REF=* .stepname.ddname REF=* .stepname.procstepname.ddname]) </pre>	<p>K Identifies copy modification patterns and table reference character.</p> <p>K Identifies group of mass storage volumes.</p> <p>K Limits the number of logical records you want included in the output data set.</p> <p>K Specifies the name of a TPROCESS macro which defines a destination queue for messages received by means of TCAM.</p> <p>K Requests channel separation.</p> <p>K Assigns space on a direct access volume for a new data set.</p> <p>K Assigns specific tracks on a direct access volume for a new data set.</p> <p>K Assigns space on a direct access volume for a new data set. Data sets share cylinders.</p> <p>K Requests part of the space on a direct access volume assigned earlier in the job.</p> <p>K Defines generalized subsystem data set.</p> <p>K Routes a data set through the output stream. For classname, assign A-Z or 0-9.</p> <p>K Indicates that an RTAM device is in use.</p> <p>K Requests a special character set for a 1403 printer.</p> <p>K Provides the system with unit information.</p> <p>K Provides the system with volume information. REF=dsname, *.stepname.ddname, and *.stepname.procstepname.ddname are invalid for VSAM components and clusters.</p>	
<p>Legend:</p> <p>P Positional parameter.</p> <p>K Keyword parameter.</p> <p>{ } Choose one.</p> <p>[] Enclosing subparameter, indicates that subparameter is optional, if more than one line is enclosed, choose one or more.</p> <p>[] Enclosing entire parameter, indicates that parameter may be optional, depending on what type of data set you are defining.</p> <p>1 Valid for 3800 only.</p> <p>2 Invalid for VSAM components and clusters.</p>				

- special characters**
- { } use 11
- [] use 11
- ... use 11
- * parameter on DD statement 74
 - coding BLKSIZE subparameter 75
 - coding BUFNO subparameter 75
 - examples of 75
- * in the PCM parameter 42
- * subparameter in the RESTART parameter 35
- *** 14
- *.ddname
 - in the DCB parameter 94,96
 - in the DSNAMES parameter 158-159
- *.stepname.ddname
 - in the DCB parameter 94,96
 - in the DSNAMES parameter 158-159
 - in the PGM parameter 42-44
- *.stepname.proclistname.ddname
 - in the DCB parameter 94,96
 - in the DSNAMES parameter 158-159
 - in the PGM parameter 42,44
- /* use 206
- //* use 205

- ABEND dumps 70
- ABSTR subparameter in the SPACE parameter 178
- accessing messages received through TCAM 173
- accounting information
 - (see accounting information parameter and ACCT parameter)
- accounting information parameter on JOB statement 19
 - continuing 19
 - example of 20
 - format of 19
 - rules for coding 19
 - special characters in 19
- ACCT parameter on EXEC statement 46
 - examples of 46
 - format of 46
 - overriding the 46
 - rules for coding 46
 - special characters in 46
- adding parameters to
 - DD statements in cataloged procedures 60
 - EXEC statements in cataloged procedures 40
- address, unit 190
- address subparameter in the SPACE parameter 177
- ADDRSPC parameter on the EXEC statement 48
 - default 48
 - examples of 48
 - format of 48
 - rules for coding 48
- ADDRSPC parameter on the JOB statement 22
 - default 22
 - examples of 22
 - format of 22
 - rules for coding 22
- AFF parameter on DD statement 80-81
 - examples of 81
 - format of 80
 - requesting channel separation 80
 - rules for coding 80
- affinity
 - channel (see channel separation)
 - unit (see UNIT parameter)
 - volume (see VOLUME parameter)
- allocating space on a direct access device 177-180
- allocating space for data sets to share cylinders 182-184
- AL subparameter 165,219
- ALIGN subparameter of FCB parameter allocation 161
- alphanumeric character set 16
- ALX subparameter in the SPACE parameter 177
- American National Standard labels 165
- AMORG, AMP subparameter for VSAM 82
- AMP parameter on DD statement 82
 - default 82
 - examples of 85
 - format of 82
 - rules for coding 84
- ANSI printer control characters in RECFM subparameter 107,116,141
- ANSI tape labels 165-167
- apostrophes
 - data set name in 16
 - purpose 16
- appendixes 211-235
- ASCII magnetic tape
 - (see DCB parameters)
 - (see LABEL parameter)
- associated data sets 216
- attributes, DCB (see DCB subparameters)
- AUL subparameter in the LABEL parameter 165
- automatic checkpoint restart (see restart facilities)
- automatic restart
 - (see automatic checkpoint restart and automatic step restart)
- automatic step restart (see restart facilities)
- automatic volume recognition (AVR)
 - channel separation requests 174
 - specifying a group name 193
- average block length
 - in SPACE parameter 177
 - in SPLIT parameter 182
 - in SUBALLOC parameter 185
- AVR (see automatic volume recognition)

- backward reference
 - to a concatenation 15
 - in DCB parameter 15
 - in DSNAMES parameter 15
 - in PGM parameter 15
 - in VOLUME parameter 15
 - with deferred restart (see restart facilities)
- BFALN, DCB subparameter
 - for BDAM 98
 - for BISAM 102
 - for BPAM 104
 - for BSAM 108
 - for EXCP 121
 - for QISAM 128
 - for QSAM 133
- BFTEK, DCB subparameter
 - for BDAM 98
 - for BSAM 108
 - for BTAM 119
 - for EXCP 121
 - for QSAM 133
- BISAM data set (see indexed sequential data set)

blank
 purpose 12,13
 BLKSIZE, DCB subparameter
 coded with
 * parameter 74,75
 DATA parameter 76,77
 DDNAME parameter 149,150
 for BDAM 98
 for BPAM 104
 for BSAM 104
 for QISAM 128
 for QSAM 134
 for TCAM 144
 block length subparameter
 in SPACE parameter 176
 in SPLIT parameter 181,182
 in SUBALLOC parameter 184
 blocks, directory, in a BPAM data set (see directory)
 blocks to be searched (see LIMCT subparameter)
 blocksize (see BLKSIZE subparameter)
 BLP subparameter in the LABEL parameter 165
 braces, use 11
 brackets, use 11
 buffers
 boundary (see BFALN subparameter)
 length of (see BUFL subparameter)
 number of (see BUFNO subparameter)
 number of, for
 all lines (see BUFSIZE subparameter)
 one line (see BUFMAX subparameter)
 receiving operation (see BUFIN subparameter)
 sending operation (see BUFOUT subparameter)
 offset (see BUFOFF subparameter)
 type (see BFTEK subparameter)
 BUFIN, DCB subparameter for TCAM 144
 BUFL, DCB subparameter
 for BDAM 99
 for BISAM 102
 for BPAM 104
 for BSAM 109
 for EXCP 121
 for QISAM 129
 for QSAM 134
 for TCAM 144
 BUFMAX, DCB subparameter for TCAM 145
 BUFND, AMP subparameter for VSAM 82
 BUFNI, AMP subparameter for VSAM 82
 BUFNO, DCB subparameter
 coded with
 * parameter 75
 DATA parameter 76
 DDNAME parameter 149
 for BDAM 99
 for BISAM 102
 for BPAM 105
 for BSAM 109
 for BTAM 119
 for EXCP 121
 for QISAM 129
 for QSAM 134
 BUFOFF, DCB subparameter
 for BSAM 109
 for QSAM 135
 BUFOUT, DCB subparameter for TCAM 145
 BUFSIZE, DCB subparameter for TCAM 145
 BUFSP, AMP subparameter for VSAM 83
 BURST parameter on DD statement
 examples of 86
 rules for coding 86
 bypass label processing (see BLP subparameter)
 bypassing I/O operations on a data set (see DUMMY
 parameter)
 bypassing a job step (see COND parameter)
 CANCEL COMMAND 202
 cataloged procedure
 calling 45
 overriding parameters on a DD statement 45
 EXEC statement
 adding parameters to 40
 overriding parameters to 40
 CATLG subparameter in the DISP parameter 151
 channel affinity (see AFF parameter)
 channel separation 80
 character arrangement tables 87,232
 character set
 alphameric 16
 national 16
 special 16
 character set for output data set 190
 CHARS parameter on DD statement 87
 character arrangement tables 87,232
 example of 87
 format of 87
 rules for coding 87
 checkid subparameter in the RESTART parameter 35
 special characters in 35
 checkpoint at end-of-volume (see CHKPT
 parameter)
 checkpoint data set 72
 checkpoint/restart facilities
 checkid 35
 checkpoint data set 72,73
 checkpoint restart 32-33,35-36,54-55,72-73
 deferred checkpoint restart 32-33,54-55
 deferred step restart 32-33,54-55
 RD parameter on EXEC statement 54-55
 RD parameter on JOB statement 32-33
 RESTART parameter on JOB statement 35-36
 SYSCHK DD statement 72-73
 CHKPT macro instruction 32-33,54-55
 CHKPT parameter on DD statement 89
 examples of 90
 format of 89
 rules for coding 89
 class
 job 23
 message 27
 system output 189
 CLASS parameter on JOB statement 23
 assigning a job class 23
 default 23
 examples of 23
 format of 23
 rules for coding 23
 classnames
 for output streams 189
 CODE, DCB subparameter
 for BSAM 110
 for EXCP 122
 for QSAM 135
 coding special characters 16
 comma, purpose 11
 command statement 202-204
 commands 202
 example of 204
 format of 202
 rules for coding 203
 commands, operator 202
 comment statement 205
 example of 205
 format of 205
 rules for coding 205
 comments field 12,14
 continuation of 14
 COMPACT parameter on DD statement
 examples of 91

rules for coding 91
 completing information in
 a data control block 94
 completion codes
 use of 24,49-50
 concatenated data set 15
 example of 16
 concatenation
 of data sets 15
 of private libraries 63-65
 COND parameter on EXEC statement 49
 examples of 50
 format of 49
 overriding 50
 rules for coding 50
 use of 49
 COND parameter on JOB statement 24
 examples of 25
 format of 24
 rules for coding 24
 use of 24
 conditional disposition of a data set 152-154
 CATLG 152
 KEEP 152
 UNCATLG 152
 conditional execution
 of a job 24
 of a job step 49
 CONTIG subparameter in the SPACE parameter 177
 continuing control statements 13
 comments field 14
 operand field 14
 control volume 214
 COPIES parameter on DD statement 92
 example of 93
 format of 92
 rules for coding 93
 used with impact printer 92,93
 used with 3800 printer 92,93
 CPU time limit
 on EXEC statement 57
 on JOB statement 37-38
 creating data sets 213-216
 nontemporary 213
 temporary 215
 CROPS, AMP subparameter for VSAM 83
 CYL subparameter
 in SPACE parameter 176
 in SPLIT parameter 181
 in SUBALLOC parameter 184
 CYLOFL, DCB subparameter for QISAM 129

 data control block
 completing the 94-95
 data definition statement
 (see DD statement)
 data in the input stream
 defining 74-77
 DATA parameter on DD statement 76-77
 coding BLKSIZE subparameter 76
 coding BUFNO subparameter 76
 examples of 76
 format of 76
 rules for coding 76
 data set
 creating a (see creating data sets)
 retrieving a (see retrieving data sets)
 data set in the input stream
 defining a 74-77
 data set label
 completing the data control block 94-96
 copying attributes from a 95
 data set name 158-160

 in apostrophes 159
 copying name from earlier DD statement 158
 nontemporary 158
 qualified 158-160
 temporary 158
 unqualified 158-160
 data set organization (see DSORG subparameter)
 DATETIME macro (see RESERVE subparameter)
 DCB
 (see data control block)
 DCB attributes 98-148
 DCB macro instruction
 completing the data control block 94-97
 DCB parameter on DD statement 94-97
 backward references to 96
 coded on
 JOBLIB DD statement 64
 STEPLIB DD statement 68
 SYSCHK DD statement 73
 coded with
 * parameter 75
 DATA parameter 76
 DDNAME parameter 149
 DUMMY parameter 78
 SYSOUT parameter 187
 completing the data control block 94-96
 copying information from
 data set label 95
 earlier DD statement 96
 examples of 96
 format of 94
 overriding subparameters in the 95-96
 rules for coding 94-96
 subparameters
 for BDAM 98-101
 for BISAM 102-103
 for BPAM 104-107
 for BSAM 108-118
 for BTAM 119-120
 for EXCP 121-126
 for GAM 127
 for QISAM 128-131
 for QSAM 133-143
 for TCAM 144-148
 DD statement 59-61
 examples of 60
 format of 59
 keyword parameters on 60
 overriding parameters on 60
 positional parameters on 60
 rules for coding 59
 ddname
 assigning a 59
 examples of 60
 qualified (see *.ddname)
 special 59
 DDNAME parameter on DD statement 149-150
 coded with
 BLKSIZE subparameter 149
 BUFNO subparameter 149
 examples of 150
 format of 149
 rules for coding 149
 dedicated data set, using 216
 default for
 CLASS parameter 23
 CPU time limit 37,58
 MSGCLASS parameter 27
 MSGLEVEL parameter 28
 output class for system messages 27
 PRTY parameter 31
 REGION parameter 34,56
 region size 34,56

UNIT parameter 192
 TIME parameter 37,57
 wait-state time limit 37,58
 DEFER subparameter in the UNIT parameter 192
 deferred checkpoint restart (see RESTART parameter)
 deferred mounting of volumes (see DEFER subparameter)
 deferred restart (see RESTART parameter)
 deferred step restart (see RESTART parameter)
 defining restart
 on EXEC statement (see RD parameter)
 on JOB statement (see RD parameter)
 DELETE subparameter in the DISP parameter 152
 delimiter statement 206
 * parameter 206
 DATA parameter 206
 example of 206
 format of 206
 rules for coding 206
 delimiter other than /* 155
 DEN, DCB subparameter
 for BSAM 110
 for EXCP 122-123
 for QSAM 136
 DEST parameter on DD statement
 examples of
 rules for coding
 device type list 232-235
 DIAGNS, DCB subparameter
 for BDAM 99
 for BISAM 103
 for BPAM 105
 for BSAM 111
 for BTAM 119
 for EXCP 123
 for GAM 127
 for QISAM 129
 for QSAM 136
 DISP parameter on DD statement 152
 coded on
 JOB LIB DD statement 63-65
 STEPLIB DD statement 68
 SYSABEND DD statement 70
 SYSCHK DD statement 72
 SYSUDUMP DD statement 70
 conditional disposition subparameter 152
 disposition subparameter 152
 examples of 154
 format of 152
 rules for coding 153
 disposition of a data set 152-154
 CATLG 152-153
 conditional disposition 152-153
 DELETE 152-153
 KEEP 152-153
 PASS 152-153
 UNCATLG 152-153
 DLM parameter 155
 example of 156
 format of 155
 DSID parameter on DD statements 157
 example of 157
 format of 157
 rules for coding 157
 DSN parameter on DD statement (see DSNAME parameter)
 DSNAME parameter on DD statement 158-160
 backward references 158-160
 coded on
 JOB LIB DD statement 63
 STEPLIB DD statement 68
 SYSABEND DD statement 70
 SYSCHK DD statement 72
 SYSUDUMP DD statement 70
 copying name from earlier DD statement 216
 examples of 159
 format of 158
 name in apostrophes 216
 nontemporary data set names 213
 rules for coding 159
 special characters in 159
 temporary data set names 215
 DSORG, DCB subparameter
 for BDAM 99
 for BISAM 103
 for BPAM 105
 for BSAM 112
 for BTAM 120
 for EXCP 123
 for GAM 127
 for QISAM 129
 for QSAM 137
 dummy data set 78
 (see also NULLFILE)
 DUMMY parameter on DD statement 78-79
 examples of 79
 format of 78
 nullifying 78
 rules for coding 78
 dump, abnormal termination 70
 storing the 70
 writing to unit record 70

 ellipsis
 use 12
 EROPT, DCB subparameter for QSAM 137
 error option (see EROPT subparameter)
 EVEN subparameter in the COND parameter 49-50
 EXEC statement 40-41
 examples of 41
 fields in 40
 format of 40
 keyword parameters on 40
 positional parameters on 40
 rules for coding 40
 execute statement (see EXEC statement)
 execution
 of a cataloged procedure 45
 of a processing program 42-44
 EXPDT subparameter in the LABEL parameter 166
 expiration date 165

 FCB parameter 161
 examples of 162
 image identifier 161
 requesting alignment of forms 161
 requesting high-density dump 161,162
 rules for coding 161
 fields
 comments 12
 examples of 12
 name 12
 operand 12
 operation 12
 fixed-length record (see RECFM subparameter)
 FLASH parameter on DD statement 163
 example of 163
 format of 163
 rules for coding 163
 FOLD subparameter in the UCS parameter 190-191
 form number subparameter in the SYSOUT parameter 187
 format of
 command statement 202
 comment statement 205
 DD statement 59

- delimiter statement 206
- EXEC statement 40
- JOB statement 17
- null statement 207
- PEND statement 208
- PROC statement 209
- forms control image (see generation data set)
- FRID, DCB subparameter for BSAM 112
- generation number, relative 214
- GNCP, DCB subparameter for GAM 127
- group name subparameter in the UNIT parameter 192
- high-density dump 70,71,161,162
- HOLD subparameter in the TYPRUN parameter 39
- holding a job for execution 39
- identifying associated data sets 216
- identifying the data set 213-221
- IEFBR14 program 44
- IN subparameter in the LABEL parameter 166
- incremental quantity (see secondary quantity)
- indexed sequential data set
 - area arrangement of 214,216,228
 - creating 213-216
 - example 160
 - lengthening 227
 - name
 - nontemporary 213-214
 - temporary 215-216
 - retrieving 213-216,227
 - example 160
- input/output macro instructions (see GNCP subparameter)
- input data set
 - concatenating 15
 - identifying the data set 213-221
 - IN subparameter 166
 - specifying
 - unit information (see UNIT parameter)
 - volume information (see VOLUME parameter)
 - conditional disposition of (see COND parameter)
 - disposition of (see DISP parameter)
- input stream
 - defining data in the 74,77
- installation requirements 19
- in-stream procedures
 - calling 45
- ISAM data set (see indexed sequential data set)
- job priority 31
- job class 23
 - default 23
 - priority 23
- job library 63-65
- JOB statement 17-18
 - examples of 18
 - fields in 17
 - format of 17
 - keyword parameters on 17
 - positional parameters on 17
- job step 40
- JOBCAT DD statement 62
- jobclass subparameter in the CLASS parameter 23
- JOBLIB DD statement 63-65 (see also STEPLIB)
 - concatenating private libraries 64
 - examples of 65
 - parameters to code when
 - cataloged 63
 - not cataloged 64
- rules for coding 63
- jobname 17
 - assigning a 17
 - examples of 18
- KEEP subparameter in the DISP parameter 152
- KEYLEN, DCB subparameter
 - for BDAM 100
 - for BPAM 105
 - for BSAM 113
 - for EXCP 124
 - for QISAM 130
- keyword parameters
 - on DD statement 60
 - on EXEC statement 40
 - on JOB statement 17
 - rules for coding 13
- LABEL parameter on DD statement 165-168
 - coded on SYSCHK DD statement 72
 - data set sequence number subparameter 165
 - examples of 167
 - EXPDT subparameter 166
 - format of 165
 - IN subparameter 166
 - label type subparameter 165
 - OUT subparameter 166
 - PASSWORD subparameter 166
 - RETPD subparameter 166
 - rules for coding 166
- label types 165
- labels
 - data set 165
 - direct access 165
 - nonstandard (NSL) 165
 - standard (SL) 165
 - standard and user (SUL) 165
 - tape 165
- libraries, concatenating private (see JOBLIB and STEPLIB)
- library
 - private 44
 - for a job 63-65
 - for a step 67-69
 - system 44
 - temporary 43
- LIMCT, DCB subparameter for BDAM 100
- limit for output records 172
- LOG command 202
- logical record (see LRECL subparameter)
- LRECL, DCB subparameter
 - for BPAM 106
 - for BSAM 113
 - for QISAM 130
 - for QSAM 138
 - for TCAM 145
- LTM subparameter on the LABEL parameter 165
- message queue records (see THRESH subparameter)
- MOD subparameter in the DISP parameter 152
- MODE, DCB subparameter
 - for BSAM 113
 - for EXCP 124
 - for QSAM 139
- mode for card reader/punch (see MODE subparameter)
- MODIFY command 202
- MODIFY parameter on DD statement 169
 - example of 169
 - format of 169
 - rules for coding 169
 - table reference character 169
- MOUNT command 202

mounting
deferred 192,193,194
parallel 192,193
MPROFILE parameter on JOB statement 26
MSGCLASS parameter on JOB statement 27
assigning an output class 27
coded with SYSOUT parameter 27
default 27
examples of 27
format of 27
rules for coding 27
MSGLEVEL parameter on JOB statement 28
default 28
examples of 29
format of 28
rules for coding 28
MSVGP parameter on DD statement 170
example of 171
format of 170
rules for coding 170
MXIG subparameter in the SPACE parameter 177

name field 12
example of 12
national character set 16
NC subparameter in the RD parameter
on EXEC statement 54
on JOB statement 32
NCP, DCB subparameter
for BISAM 103
for BPAM 106
for BSAM 114
NEW subparameter in the DISP parameter 152
NL subparameter in the LABEL parameter 165
nonspecific volume request 201
for direct access volume 201
satisfying a 201
for tape volume 201
nonstandard labels 165
label type subparameter 165
nontemporary data set, creating 213
NOPWREAD subparameter in the LABEL
parameter 165
NR subparameter in the RD parameter
on EXEC statement 54
on JOB statement 32
NSL subparameter in the LABEL parameter 165
NTM, DCB subparameter for QISAM 130
null statement 207
example of 207
format of 207
NULLFILE (see DUMMY parameter)

OLD subparameter in the DISP parameter 152
ONLY subparameter in the COND parameter
on EXEC statement 49,50
OPEN/CLOSE/EOV trace option (see DIAGNS
subparameter)
operand field 12
blank 14
example of 12
keyword parameters 13
positional parameters 13
subparameters 13
operation field 12
operator commands 200-201
operator subparameter in the COND parameter
on EXEC statement 49-50
on JOB statement 24
OPTCD, AMP subparameter for VSAM 83
OPTCD, DCB subparameter
for BDAM 100
for BPAM 106
for BSAM 115
for EXCP 124
for QISAM 130
for QSAM 139
for TCAM 146
optional services (see OPTCD subparameter)
OUT subparameter in the LABEL parameter 166
OUTLIM parameter 172
coded with SYSOUT parameter 172
determining the output limit 172
example 172
rules for coding 172
output of
allocation messages 28
allocation recovery messages 28
disposition messages 28
job control statements 28
output class
for output data set 187
for system messages 27
output class subparameter in the MSGCLASS
parameter 27
output data set
allocating space for 176
OUT subparameter 166
printed using UCS feature
(see UNIT parameter)
(see VOLUME parameter)
routed through output stream 187-188
specifying
conditional disposition (see COND parameter)
disposition (see DISP parameter)
status (see DISP parameter)
output stream
routing data sets through the 187-188
output writer 187-188

P subparameter in the UNIT parameter 192
parallel mounting 192,193
parameter 11
parentheses
to enclose a subparameter list 13
inclusion in variables 13
PARM parameter on EXEC statement 52,53
examples of 53
format of 52
overriding the 52
rules for coding 52
special characters in 52
partition 34,56
partitioned data set
concatenating 15
executing programs in a 45
name
nontemporary 213
temporary 215
retrieving a member of 213
PASS subparameter in the DISP parameter 152
passing variable information to a program 52-53
password protection 167
PASSWORD subparameter in the LABEL
parameter 168
PCI, DCB subparameter for TCAM 146
PEND statement 208
PGM parameter on EXEC statement 42-44
backward references 42
examples of 42
executing programs from
private library 44
system library 44
temporary library 43
format of 42

positional parameters
 on DD statement 60
 on EXEC statement 40
 on JOB statement 17
 rules for coding 13

postponing definition of a data set (see DDNAME parameter)

primary quantity
 in SPACE parameter 176-180
 in SPLIT parameter 181-182
 in SUBALLOC parameter 184-185

priority
 job 31
 job class 23

priority parameter (see PRTY parameter)

private libraries 44,63,67
 concatenating 63,67
 defining 63,67
 executing programs from 42-44

PRIVATE subparameter in the VOLUME parameter 197

PROC parameter on EXEC statement 45
 examples of 45
 format of 45

PROC statement 209-210
 examples of 210
 format of 209
 rules for coding 209

procedure
 (see cataloged procedure; instream procedure)

PROFILE parameter
 JOB statement 30

program, calling a 42-43

program controlled interruption (see PCI subparameter)

program name
 subparameter in the SYSOUT parameter 187

programmer's name parameter on JOB statement 21
 examples of 21
 format of 21
 rules for coding 21
 special characters in 21

PRTSP, DCB subparameter
 for BSAM 115
 for EXCP 125
 for QSAM 140

PRTY parameter on JOB statement 31
 default 31
 examples of 31
 format of 31
 rules for coding 31

QISAM data set
 DCB subparameter 128-132
 (see also ISAM data set)

QNAME parameter on the DD statement 170
 example of 173
 format of 173
 rules for coding 173

qualified name
 assigning a 213

R subparameter in the RD parameter
 on EXEC statement 54
 on JOB statement 32

RD parameter on EXEC statement
 defining restart 54
 examples of 55
 format of 54
 overriding the 54
 restart facilities 54
 rules for coding 54

RD parameter on JOB statement
 defining restart 32
 examples of 33
 format of 32
 overriding the 32
 restart facilities 32
 rules for coding 32

REAL subparameter in the ADDRSP parameter
 on the EXEC statement 48
 on the JOB statement 22

real address space 22,48

real storage 22,48

RECFM, AMP subparameter for VSAM 84

RECFM, DCB subparameter
 for BDAM 101
 for BPAM 107
 for BSAM 116
 for QISAM 131
 for QSAM 141
 for TCAM 147

record format (see RECFM subparameter)

record key position (see RKP subparameter)

recording technique for seven-track tape (see TRTCH subparameter)

record length (see LRECL subparameter)

REF subparameter in the VOLUME parameter 196,199

references, backward 15

region size 34,56

REGION parameter on EXEC statement 56
 default 56
 examples of 56
 format of 56
 overriding the 56
 rules for coding 56

REGION parameter on JOB statement 34
 default 34
 examples of 34
 format of 34
 rules for coding 34

relational operators in the COND parameter
 on the EXEC statement 49-51
 on the JOB statement 24

RELEASE command 202

releasing unused space (see RLSE)

remote job entry
 restriction on use of BUFNO subparameter
 with * parameter 74
 with DATA parameter 76
 with DDNAME parameter 149

removable volume 200-201

REPLY command 202

requesting channel separation 174

requesting copies of an output data set 92

RESET command 203

RESERVE, DCB subparameter for TCAM 147

reserved volume 200-201

restart definition (RD parameter) 195,196
 on EXEC statement 54-55
 on JOB statement 32-33

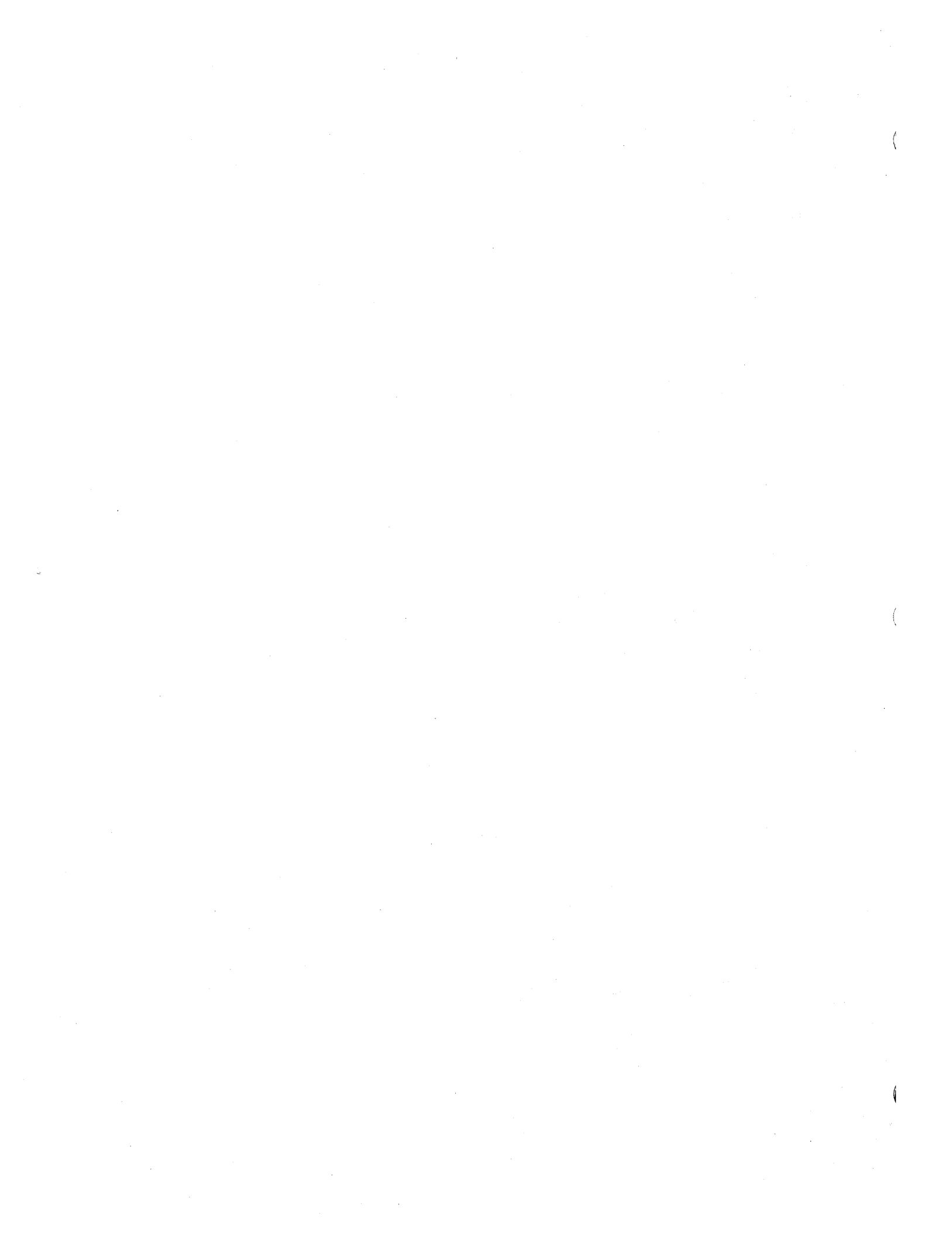
restart facilities
 RD parameter on EXEC statement 54-55
 RD parameter on JOB statement 32-33
 RESTART parameter on JOB statement 35-36
 RESTART parameter on JOB statement 35-36
 examples of 36
 format of 35
 rules for coding 35
 when defining generation data set 36
 when making backward reference 36

RETAIN subparameter in the VOLUME parameter 196,197

retention period 167
RETPD subparameter in the LABEL parameter 166
retrieving data sets
 member of partitioned data set 214
return code test 24,49-51
RKP, DCB subparameter for QISAM 132
RLSE subparameter in the SPACE parameter 176
RNC subparameter in the RD parameter
 on EXEC statement 54
 on JOB statement 32
ROLL parameter
 on EXEC statement 6
 on JOB statement 6
ROUND subparameter in the SPACE parameter 177

SCAN, subparameter in the TYPRUN parameter 39
scanning JCL for syntax errors 39
secondary quantity
 in SPACE parameter 176
 in SPLIT parameter 181
 in SUBALLOC parameter 184
SEP parameter in DD statement 174
 examples of 175
 format of 174
 requesting channel separation 174
 rules for coding 174
SEP subparameter in the UNIT parameter 192,194
separation
 channel 71,174
 unit 192,194
SEQUENCE macro (see RESERVE subparameter)
sequence number
 data set 164
 volume 196,197
sequence number in columns 73-80 13
sequential data set
 concatenating 15
SER subparameter in the VOLUME parameter 198,200
SET command 203
sharing
 cylinders (see SPLIT parameter)
 data set (see DISP parameter)
shortening processing time 174
SHR subparameter in the DISP parameter 152
SL subparameter in the LABEL parameter 165
SPACE parameter on DD statement 176-180
 (see also SPLIT; SUBALLOC)
 coded on
 SYSABEND DD statement 70
 SYSUDUMP DD statement 70
 examples of 179
 format of 176
 rules for coding 177
space on a printer (see PRTSP subparameter)
special character set
 with UCS parameter 190
 using 16
special ddnames 59
specific volume request 200
SPLIT parameter on DD statement
 (see also SPACE; SUBALLOC)
 coded on
 SYSABEND DD statement 70
 SYSUDUMP DD statement 70
 examples of 182
 format of 181
 rules for coding 182
STACK, DCB subparameter 117,125,142
stacker bin (see STACK subparameter)
 for BSAM 117
 for EXCP 125
 for QSAM 142
START command 203
step restart
 automatic 32,54
 deferred 35,36
STEPCAT DD statement 66
STEPLIB DD statement 67-69
 (see also JOBLIB)
 concatenating private libraries 67
 examples of 68
 parameters to code when
 cataloged 68
 not cataloged or not passed 68
 passed 68
 rules for coding 67
stepname
 assigning a 40
 examples of 41
STOP command 203
stream, input, data sets in the 74-77
stream, output, routing data sets through the 187-188
STRNO, AMP subparameter for VSAM 84
SUBALLOC parameter on DD statement 184-185
 (see also SPACE; SPLIT)
 coded on
 SYSABEND DD statement 70
 SYSUDUMP DD statement 70
 examples of 185
 format of 184
 rules for coding 185
suballocation (see SUBALLOC parameter)
SUBSYS parameter 186
SUL subparameter in the LABEL parameter 165
suppressing
 CHKPT macro instruction 32-33,54-55
 automatic restarts 32-33,54-55
SYNAD, AMP subparameter for VSAM 84
SYSABEND DD statement 70-71
 (see also SYSUDUMP)
 examples of 70
 high-density dump 70,71
 storing the dump 70
 writing the dump to unit record device 70
SYSCHK DD statement 72-73
 examples of 73
 parameters to code when
 cataloged 72
 not cataloged 73
 rules for coding 72
SYSOUT parameter on DD statement 187-188
 coded on
 SYSABEND DD statement 70
 SYSUDUMP DD statement 70
 examples of 188
 format of 187
 rules for coding 187
 specifying classname 187
 specifying DCB parameter 187
 specifying form number 187
 specifying program name 187
system library 43
System Management Facilities
 with TIME parameter 37-38,58
system messages
 output class 27
SYSUDUMP DD statement
 (see also SYSABEND)
 examples of 70-71
 high-density dump 70,71
 storing the dump 70
 writing the dump to unit record device 70
SYS1.LINKLIB 44

table reference character	169		
tape density (see DEN subparameter)			
tape labels, ANSI	165-167		
TCAM (see Telecommunications Access Method)			
Telecommunications Access Method (TCAM)			
DCB subparameter	145-147		
temporary data set			
creating	215-216		
temporary library	43		
THRESH, DCB subparameter	147		
time limit			
wait state	58		
TIME parameter on EXEC statement	57-58		
CPU time limit	57		
default	58		
effect of JOB limit	57		
eliminating timing	58		
examples of	58		
format of	57		
overriding the	57		
rules for coding	57		
wait-state time limit	58		
TIME parameter on JOB statement	37-38		
CPU time limit	37		
default	37		
effect of JOB limit	37		
eliminating timing	37		
examples of	38		
format of	37		
rules for coding	37		
wait-state time limit	37		
1440	37		
timing			
CPU	37,57		
eliminating	37,58		
TRACE, AMP subparameter for VSAM	84		
tracks for cylinder index (see NTM subparameter)			
tracks for overflow (see CYLOFL subparameter)			
tracks to the searched (see LIMCT subparameter)			
TRK subparameter			
in SPACE parameter	176		
in SUBALLOC parameter	184		
TRTCH, DCB subparameter			
for BSAM	117		
for EXCP	125		
for QSAM	142		
TYPRUN parameter on JOB statement	39		
example of	39		
format of	39		
rules for coding	39		
UCS parameter on DD statement	190-191		
examples of	191		
format of	190		
identifying character set	190		
requesting			
fold mode	190		
operator verification	190		
rules for coding	190		
special character sets	190		
UNCATLG subparameter in the DISP parameter	152		
unit address	192		
unit affinity	193		
unit count subparameter in the UNIT parameter	192		
UNIT parameter on DD statement	192-195		
coded on			
JOB LIB DD statement	64		
STEPLIB DD statement	68		
SYSABEND DD statement	70		
SYSCHK DD statement	72,73		
SYSUDUMP DD statement	70		
examples of	195		
format of	192		
identifying the device	192,193		
providing unit information	192-193		
rules for coding	193		
specifying			
deferred mounting	194		
parallel mounting	192		
unit affinity	194		
unit count	192		
unit separation	194		
with suballocation	185		
unit record devices			
writing dumps to	70-71		
unit separation	194		
universal character set (see UCS)			
UNLOAD command	203		
unqualified name, assigning	158-159		
V format	84		
VARY command	203		
VERIFY subparameter			
of FCB parameter	161		
of UCS parameter	190		
VIRT subparameter in the ADDRSPC parameter			
on the EXEC statement	48		
on the JOB statement	22		
virtual address space	22,48		
virtual storage	22,48		
VOL parameter on DD statement (see VOLUME parameter)			
volume (see VOLUME parameter)			
volume count subparameter in the VOLUME parameter	196		
VOLUME parameter on DD statement	196-201		
backward reference	193-199		
coded on			
JOB LIB DD statement	64		
STEPLIB DD statement	68		
SYSABEND DD statement	70		
SYSCHK DD statement	72-73		
SYSUDUMP DD statement	70		
examples of	200		
format of	196		
providing volume information	196-201		
referring to specific request	197		
rules for coding	197		
specifying			
PRIVATE subparameter	197		
RETAIN subparameter	197		
volume sequence number subparameter	197		
volume count subparameter	198		
with suballocation	185		
supplying serial numbers	195		
volume sequence number subparameter in the VOLUME parameter	196		
for checkpoint entry	72		
volume serial number			
for checkpoint entry	72		
special characters in	194		
VOLUME=REF			
backward references	199		
wait state time limit	37,58		
writing output messages	27,29		
XX*	14		





This Newsletter No. GN24-5628
Date March 30, 1979
Base Publication No. GC24-5099-4
File No. S370-36
(OS/VS1 Releases 6.7 and 7)
Previous Newsletters None

OS/VS1 JCL Reference

© IBM Corp. 1973, 1975, 1976, 1977, 1979

This Technical Newsletter, a part of Releases 6.7 and 7 of OS/VS1, provides replacement and additional pages for your publication. These replacement and additional pages remain in effect for subsequent OS/VS1 releases unless specifically altered. Pages to be added or replaced are:

Cover - 6	140.1, 140.2 (text rearranged only)
39, 40	197, 198
113 - 116	198.1, 198.2 (text rearranged only)
116.1, 116.2 (text rearranged only)	233, 234
139, 140	

A change to the text or to an illustration is indicated by a vertical line to the left of the change.

Summary of Amendments

- Updates OS/VS1 Release 6.7 support to reflect changes for Mass Storage System Extensions, Program Number 5740-XYG.
- The 3278-2A Display Console is added to the list of valid device types for OS/VS1 Release 7, Program Number 5652-VS1.

Note: *Please insert this page in your publication to provide a record of changes.*





International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, N. Y. 10604

IBM World Trade Americas/Far East Corporation
Town of Mount Pleasant, Route 9, North Tarrytown, N. Y., U. S. A. 10591

IBM World Trade Europe/Middle East/Africa Corporation
360 Hamilton Avenue, White Plains, N. Y., U. S. A. 10601