

Systems

OS/VS JCL Reference

**VS1 Release 2
VS2 Release 1**

IBM

Third Edition (May, 1973)

This is a reprint of GC28-0618-1 incorporating changes released in the following Technical Newsletter:

GN28-2539 (dated December 15, 1972)

This edition applies to release 2 of OS/VS1 and release 1 of OS/VS2 and to all subsequent releases until otherwise indicated in new editions or Technical Newsletters. Changes are continually made to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest *IBM System/360 and System/370 Bibliography*, GA22-6822, and the *IBM System/370 Advanced Function Bibliography*, GC20-1763, for the editions that are applicable and current.

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

A form for readers' comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Department G60, P.O. Box 6, Endicott, New York 13760. Comments become the property of IBM.

Preface

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

This publication consists of:

1. Programming notes, which contain coding conventions for coding job control language statements.
2. Descriptions of job control language statements, which include the format of each statement and the format of the parameters associated with the statement.

There is a separate section for each statement. Also, for every statement and each of its associated parameters there is a definition given, followed by a reference to the appropriate publication for a detailed explanation of the facility or service to be used. The definition and reference are in turn followed by rules for coding and at least one example of how to code the control statement or parameter.

3. Appendixes, which include additional information on some job control language facilities, including several tables for quick references.
4. Foldout charts, which show the format of JOB, EXEC, and DD statement parameters. The foldout charts appear after the index.

The following OS/MFT and MVT job control language parameters do not apply to OS/VS. If left unchanged on JCL statements, they will be checked for correct syntax by the reader/interpreter, but will be otherwise ignored by the system.

On the JOB statement:

ROLL

On the EXEC statement:

ROLL

On the DD statement:

HIARCHY subparameter of the DCB parameter

The REGION parameter for the JOB and EXEC statements has been redefined for use with virtual storage.

For OS/VS2, the following parameters have been included in this edition of the publication.

On the JOB statement:

NOTIFY (for TSO)

On the EXEC statement:

DPRTY

On the DD statement:

DYNAM (for TSO)

TERM (for TSO)

Before you use this publication, which is intended as a reference book, you must understand the concepts and terminology introduced in the prerequisite publications listed below:

Prerequisite publications:

OS/VS JCL Services, GC28-0617.

Publications to which the text refer

OS/VS Checkpoint/Restart, GC26-3784.

OS/VS Data Management for Systems Programmers, GC28-0631.

OS/VS Data Management Services, GC26-3783.

OS/VS Debugging Guide, GC24-5093.

OS/VS JCL Services, GC28-0617.

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

OS/VS Supervisor Services and Macro Instructions, GC27-6979.

OS/VS System Management Facilities (SMF), GC35-0004.

OS/VS Tape Labels, GC26-3795.

CONTENTS DIRECTORY

Programming Notes	→	Notes
JOB Statement	→	JOB
EXEC Statement	→	EXEC
DD Statement	→	DD
Command Statement	→	Command
Comment Statement	→	Comment
Delimiter Statement	→	Delimiter
Null Statement	→	Null
PEND Statement	→	PEND
PROC Statement	→	PROC
Appendixes	→	Appendix
Index	→	Index
Foldout Charts	→	Chart

Contents

Programming Notes	13
The JOB Statement	21
The Accounting Information Parameter	23
The Programmer's Name Parameter	24
The ADDRSPC Parameter	25
The CLASS Parameter	26
The COND Parameter	27
The MSGCLASS Parameter	28
The MSGLEVEL Parameter	29
The NOTIFY Parameter	31
The PRTY Parameter	32
The RD Parameter	33
The REGION Parameter	35
The RESTART Parameter	36
The TIME Parameter	38
The TYPRUN Parameter	40
The EXEC Statement	41
The PGM Parameter	43
The PROC Parameter	46
The ACCT Parameter	47
The ADDRSPC Parameter	48
The COND Parameter	49
The DPRTY Parameter	52
The PARM Parameter	54
The RD Parameter	56
The REGION Parameter	58
The TIME Parameter	59
The DD Statement	61
The JOBLIB Facility	63
The STEPLIB Facility	66
The SYSABEND and SYSUDUMP Facilities	69
The SYSCHK Facility	71
The * Parameter	73
The DATA Parameter	75
The DUMMY Parameter	77
The DYNAM Parameter	79
The AFF Parameter	80
The COPIES Parameter	82
The DCB Parameter	83
The DCB Subparameters for BDAM	87
The DCB Subparameters for BISAM	91
The DCB Subparameters for BPAM	93
The DCB Subparameters for BSAM	97
The DCB Subparameters for BTAM	107
The DCB Subparameters for EXCP	109
The DCB Subparameters for GAM	115
The DCB Subparameters for QISAM	117
The DCB Subparameters for QSAM	121
The DCB Subparameters for TCAM	131
The DDNAME Parameter	135
The DEST Parameter	136.1
The DISP Parameter	137
The DLM Parameter	140
The DSNAME Parameter	141
The FCB Parameter	143
The HOLD Parameter	144
The LABEL Parameter	144.1
The OUTLIM Parameter	147
The QNAME Parameter	148
The SEP Parameter	149
The SPACE Parameter	150
The SPLIT Parameter	153
The SUBALLOC Parameter	155

The SYSOUT Parameter	157
The TERM Parameter (For OS/VS1)	159
The TERM Parameter (For OS/VS2)	160
The UCS Parameter	160.1
The UNIT Parameter	162
The VOLUME Parameter	165
The Command Statement	171
The Comment Statement	173
The Delimiter Statement	175
The Null Statement	177
The PEND Statement	179
The PROC Statement	181
Appendix A: Identifying the Data Set to the System	183
Specifying the DDNAME Parameter	183
When You Code the DDNAME Parameter	183
Specifying the DSNNAME Parameter	185
Creating or Retrieving a Nontemporary Data Set	185
Nontemporary Data Sets	185
Members of a Partitioned Data Set	186
Generations of a Generation Data Group	186
Areas of an Indexed Sequential Data Set	186
Creating or Retrieving a Temporary Data Set	186
Temporary Data Sets	187
Members of a Temporary Partitioned Data Set	187
Areas of a Temporary Indexed Sequential Data Set	187
Using a Dedicated Data Set	188
Copying the Data Set Name from an Earlier DD Statement	188
Specifying the DSNNAME Parameter in Apostrophes	188
Specifying the LABEL Parameter	189
The Data Set Sequence Number Subparameter	189
The Label Type Subparameter	190
The PASSWORD and NOPREAD Subparameters	191
The IN and OUT Subparameters	191
The RETPD and EXPDT Subparameters	192
Appendix B: Cataloged and In-Stream Procedures (Removed and not replaced)	
Appendix C: Requesting Space for a Data Set on a Direct Access Volume (Removed and not replaced)	
Appendix D: Creating and Retrieving Indexed Sequential Data Sets	205
Creating an Indexed Sequential Data Set	205
The DSNNAME Parameter	205
The UNIT Parameter	206
The VOLUME Parameter	206
The LABEL Parameter	206
The DCB Parameter	206
The DISP Parameter	206
The SPACE Parameter	207
Nonspecific Allocation Technique	207
Absolute Track Technique	207
The SEP or AFF Parameter	207
Area Arrangement of an Indexed Sequential Data Set	208
Retrieving an Indexed Sequential Data Set	208
The DSNNAME Parameter	208
The UNIT Parameter	208
The VOLUME Parameter	209
The DCB Parameter	209
The DISP Parameter	209
Example of Creating and Retrieving an Indexed Sequential Data Set	209

Appendix E: Creating and Retrieving Generation Data Sets	210
Before You Define the First Generation Data Set	210
Creating a Model Data Set Label	210
Referring the System to a Cataloged Data Set	211
Creating a Generation Data Set	211
The DSNNAME Parameter	211
The DISP Parameter	211
The UNIT Parameter	211
The VOLUME Parameter	212
The SPACE Parameter	212
The LABEL Parameter	212
The DCB Parameter	212
Retrieving a Generation Data Set	212
The DSNNAME Parameter	212
The DISP Parameter	213
The UNIT Parameter	213
The LABEL Parameter	213
The DCB Parameter	213
Resubmitting a Job for Restart	213
Example of Creating and Retrieving a Generation Data Set	214
Appendix F: Reference Tables	215

Figures

Figure 1. Control Statement Fields	14
Figure 2. Character Sets	18
Figure 3. DD Parameters for Creating a Data Set	216
Figure 4. DD Parameters for Retrieving a Data Set	217
Figure 5. DD Parameters for Extending a Data Set	218
Figure 6. DD Parameters for Retrieving or Extending an Indexed Sequential Data Set	220
Figure 7. Area Arrangement of Indexed Sequential Data Sets	221
Figure 8. Table of Mutually Exclusive DD Parameters	222
Figure 9. Disposition Processing Chart (Turnpage)	223
Figure 10. Direct Access Capacities	224
Figure 11. Track Capacities	224
Figure 12. The JOB Statement (foldout)	241
Figure 13. The EXEC Statement (foldout)	243
Figure 14. The DD Statement (foldout)	245

**Summary of Amendments
For GC28-0618-1
VS1 Release 2**

For OS/VS1 only, the following parameters
have been included in this edition of the
publication.

On the DD statement:

DEST (For RES, remote entry services)

HOLD (For RES)

TERM(For RES)

TCAM Level II

Level II of TCAM will not run under Release 2 of VS1.

**Summary of Amendments
For GC28-0618-1
VS2 Release 1**

For OS/VS2, only, the following parameters
have been included in this publication.

On the JOB statement:

NOTIFY (for TSO)

On the EXEC statement:

DPRTY

On the DD statement:

DYNAM (for TSO)

TERM (for TSO)

Notation for Defining Control Statement Parameters

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

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

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

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

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

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

For example, $\left\{ \begin{array}{l} \text{TRK} \\ \text{CYL} \\ \text{block size} \end{array} \right\}$ is part of the format description for the SPACE parameter.

When you code the SPACE parameter, you must code either TRK, CYL, or a substitute for “block size,” which would be a number.

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

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

An example of more than one item enclosed in brackets is

$\left[\begin{array}{l} \text{EXPDT=yyddd} \\ \text{RETPD=nnnn} \end{array} \right]$

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

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

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

For example, [,progrname] [,form number] is part of the format description for the SYSOUT parameter. When you code the SYSOUT parameter, you have the option of coding both “,progrname” and “,form number”, omitting both, or coding only one. The

comma enclosed in brackets with “, progname” must be coded when “,progname” is not coded but “,form number” is coded; that is, you would code: „form number.

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

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

Fields in Control Statements

Every control statement is logically divided into different fields. There are four fields -- name field, operation field, operand field, comments field -- but not all of the control statements can contain all of these fields. Figure 1 shows the fields for each statement.

Statement	Columns 1 and 2	Fields
JOB	//	name operation (JOB) operand ¹ comments ²
EXEC	//	name ¹ operation (EXEC) operand comments ¹
DD	//	name ¹ operation (DD) operand comments ¹
PROC (Cataloged)	//	name ¹ operation (PROC) operand comments ¹
PROC (in-stream)	//	name operation (PROC) operand ¹ comments ²
Procedure end	//	name ¹ operation (PEND) comments ²
Command	//	operation (command) operand comments ¹
Delimiter	/*	comments ¹
Null	//	
Statement	Columns 1, 2, 3	Field
Comment	/*	comments
¹ Optional ² Optional -- If operand(s) are not coded, comments cannot be coded. If operand(s) are coded, comments are optional.		

Figure 1. Control Statement Fields

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

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

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

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

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

Except for the comment statement, which can be coded through column 80, fields cannot be coded past column 71. If the total length of the fields will exceed 71 columns, you must

continue the fields onto one or more succeeding statements. How to continue fields is described under “Continuing Control Statements.”

Parameters in the Operand Field

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

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

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

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

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

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

Continuing Control Statements

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

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

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

To continue the operand field:

1. Interrupt the field after a complete parameter or subparameter, including the comma that follows it, at or before column 71.
2. Comments can be included by following the interrupted field with at least one blank.

3. Optionally, code any nonblank character in column 72. (The nonblank character in column 72 is required only when you are continuing a comments field.) If you do not code a character in column 72 when continuing the operand field, the system treats the next statement as a continuation statement as long as you follow the conventions outlined in items 4 and 5.
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. If you leave the statement blank after column 2 or if you begin coding after column 16, the system assumes that no other operands are present and treats any characters you code as a comment field.

To continue the comments field:

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

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

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

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

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

Concatenating Data Sets

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

If concatenated data sets have unlike characteristics, e.g., the device types, block lengths, or record formats differ, the DCBOFLGS field of the data control block must be modified while the program is executing. For details, refer to *OS/VS Data Management Services Guide*, GC26-3783.

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

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

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

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

```
//INPUT      DD          DSNAME=A.B.C,DISP=(OLD,DELETE)
//          DD          DSNAME=X.Y.Z,DISP=OLD,LABEL=(,NL)
//          DD          DSNAME=ALPHA,UNIT=2314,VOLUME=SER=P12,      X
//          DD          DISP=(OLD,DELETE)
//          .
//          .
//          data
//          .
//          .
/*
```

Character Sets

Job control statements are coded using a combination of the characters from three different character sets. The contents of each of the character sets are described in Figure 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	, . / ' () * & + - =

Figure 2. Character Sets

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

These rules and the use of special characters are described next.

Using Special Characters

Special characters are used in the job control language to:

1. Delimit parameters (the comma).
2. Delimit fields (the blank).
3. Perform syntactical functions. (For example, the appearance of $\epsilon\epsilon$ as the first two characters following `DSNAME=` tells the system that a temporary data set name follows. The appearance of `/` in the `UNIT` parameter, `UNIT=293/5`, tells the system that a specific device is desired.)

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

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

1. The accounting information on the `JOB` statement. The account number and additional accounting information can contain hyphens without being enclosed in apostrophes.
2. The programmer's name on the `JOB` statement. The programmer's name can contain periods without being enclosed in apostrophes.
3. The checkid field in the `RESTART` parameter on the `JOB` statement.

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

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 -- the name, operation (JOB), operands, and comments fields.

Rules for Coding

1. Code a JOB statement for each job. Code a unique jobname in every JOB statement. The jobname must consist of 1 through 8 alphanumeric and national (#,@,\$) characters; the first character must be alphabetic or national.
2. There are two types of parameters that can be coded on the JOB statement:

Positional parameters, which must precede any keyword parameters and must be coded in the following order.

accounting information
programmer's name

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

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

ADDRSPC
CLASS
COND
MSGCLASS
MSGLEVEL
NOTIFY
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. Do not use the following names and characters as jobnames because they are keywords of the DISPLAY command, unless you enclose the name used in quotation marks.

A
CONSOLES
DSNAME
JOBNAME
N
Q
R
SPACE
STATUS
T
U

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 catalogued procedure for the input reader.

For information on how to add accounting facilities, refer to **OS/VS1 Planning and Use Guide**, GC24-5090 or **OS/VS2 Planning and Use Guide**, GC28-0600.

([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, (5-8 punch), e.g. '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, e.g., '5438,10/08/66' or
 - b. Enclose the item in apostrophes and the accounting information in parentheses, e.g., (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, e.g., '34&&8241'.

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.

The ADDRSPC Parameter

Keyword, Optional

The ADDRSPC parameter can be used to assure 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/VS JCL Services, GC28-0617.

$$\text{ADDRSPC} = \begin{cases} \text{VIRT} \\ \text{REAL} \end{cases}$$

Rules for Coding

1. For OS/VS1, code the REGION parameter to specify how much real storage you need.
For OS/VS2, code the REGION parameter to specify how much real or virtual 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 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 will be ignored and the value on the JOB statement will be used.
4. For OS/VS1, when you specify ADDRSPC=REAL, the REGION size you specify overrides the system partition size. Therefore, make the REGION size large enough to satisfy all real storage requests for a job, keeping in mind that the REGION size cannot be larger than the partition size.

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=VIRTUAL
```

The ADDRSPC parameter requests virtual storage. Since the REGION parameter is omitted, the default region size for the installation will be used.

The 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 use of the CLASS parameter, see OS/VS JCL Services, GC28-0617.

```
CLASS=jobclass
```

Rules for Coding

1. Code any alphabetic character from A through O, depending on the installation's rules for assigning a jobclass.
2. **Default:** If you omit the CLASS parameter, jobclass A 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.

The COND Parameter

Keyword, Optional

The COND parameter specifies whether a job will continue processing based on completion 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 use of the COND parameter, see *OS/VS JCL Services*, GC28-0617.

```
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.

Example of the COND Parameter

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

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

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

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

The MSGCLASS Parameter

Keyword, Optional

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

For further information on use of the MSGCLASS parameter, see OS/VS JCL Services, GC28-0617.

```
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 will be 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.

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 will default to the MSGCLASS value specified in the PARM field of the input reader procedure. The default is A unless changed by your installation.

```
//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.

The MSGLEVEL Parameter

Keyword, Optional

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

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

For further information on the MSGLEVEL parameter, see *OS/VS JCL Services*, GC28-0617.

```
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 which indicates what allocation/termination messages are to be written. 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.

The NOTIFY Parameter (For OS/VS2 with TSO)

Keyword, Optional

The NOTIFY parameter is used to request that a message be sent to a user's time sharing terminal when his background job has completed processing.

For a detailed discussion of TSO, see *OS/VS2 TSO Guide*, GC28-0644.

```
NOTIFY=user identification
```

user identification
the identification of the user to be notified.

Rules for Coding

1. Code a 1 to 7 alphameric identification; the first character must be alphabetic.
2. Code the same user identification that you specify when you start a terminal session (at LOGON).
3. If the NOTIFY parameter is coded for OS/VS1, the parameter is checked for syntax but otherwise ignored.

Example of the NOTIFY Parameter

```
//SIGN      JOB      NOTIFY=POK1
```

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

The 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 parameter on the JOB statement.) When the job is initiated, the system will convert the job's priority into a dispatching priority so that the job's tasks can compete with other tasks for use of main storage and CPU resources.

For further information on the use of the PRTY parameter, see *OS/VS JCL Services*, GC28-0617.

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.

Examples of the PRTY Parameter

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

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

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

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

The RD Parameter

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 restart.

For detailed information on the checkpoint/restart facility, refer to:

- OS/VS Checkpoint/Restart, GC26-3784.

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

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. For OS/VS1 when you code RD=R or RD=RNC, you must code MSGLEVEL=(1,0), MSGLEVEL=(1,1), or MSGLEVEL=1 on the JOB statement. For OS/VS2, you may omit the MSGLEVEL parameter.
2. If automatic step restart is permitted, i.e., when you have coded RD=R or RD=RNC, assign each job step a unique step name.
3. Code the RD parameter on EXEC statement, instead of the JOB statement, when you want to make different restart requests for each job step.
4. 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.

Examples of the RD Parameter

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

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

```
//TRY56     JOB      RD=RNC,MSGLEVEL=( 1,1 )
```

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

```
//PASS      JOB      RD=NR,MSGLEVEL=( 0,1 )
```

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

The REGION Parameter

Keyword, Optional

For OS/VS1, the REGION parameter specifies the amount of real storage to be allocated to a job.

For OS/VS2, the REGION parameter specifies the amount of real or virtual address space to be allocated to a job. The REGION parameter should be specified or the REGION size default value in the input reader procedure will be used.

For further information on the REGION parameter, see OS/VS JCL Services, GC28-0617.

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 will cause a JCL error.
2. **Default:** If you omit the REGION parameter, the REGION size default value in the input reader procedure will be used.
3. When you want to specify a different region size for each job step, code the REGION parameter on EXEC statements, instead of 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. In OS/VS1, if you code ADDRSPC=VIRTUAL with the REGION parameter, REGION will be ignored.
6. In OS/VS1, the REGION size you specify overrides the system partition size. Therefore, make the REGION size large enough to satisfy all real storage requests for a job, keeping in mind that the REGION size must be larger than the partition size.

Examples of the REGION Parameter

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

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

The 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 the following publication:

- OS/VS Checkpoint/Restart, GC26-3784.

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

*

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, i.e., 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 (i.e.,DISP=NEW,CATLG,KEEP) was coded and abnormal termination occurred), then during checkpoint restart, refer to these data sets and generation data sets by the same relative generation numbers that were used to create them.

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.

The 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 System Management Facilities feature, see OS/VS System Management Facilities (SMF), GC35-0004.

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

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, there is no CPU time limit assigned to a job; however, each job step can still be timed.
6. **System Management Facilities:** Normally, a job that exceeds the time limit specified in the TIME parameter is terminated. However, if the System Management Facilities are in use, a user exit routine can extend the time limit so that processing can continue.

Without the System Management Facilities 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 causes termination of the job.

With the System Management Facilities in use in the system, the installation determines the time limit for wait states. A job step remaining in a wait state for more than the established time limit causes termination of the job unless a user-provided exit routine extends the wait state limit for that step.

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.

JOB

The TYPRUN Parameter

Keyword, Optional

For OS/VS1 and OS/VS2 the TYPRUN parameter specifies that a job 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.

For OS/VS1 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/VS JCL Services*, GC28-0617.

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

HOLD

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

SCAN (for OS/VS1 only)

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

Rules for Coding

1. Coding TYPRUN=SCAN in OS/VS2 will cause a JCL error.

Example of the TYPRUN Parameter

Jobs UPDATE and LIST are to be submitted for execution. The job UPDATE uses a program that adds and deletes members to a library; the job LIST uses a program that lists the members to 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 DISPLAY JOBNAMES command is issued by you or the operator, the operator is notified on the console when UPDATE has completed processing; he issues a RELEASE command for LIST. The job LIST can then be selected for execution.

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.

A black circular logo with the word "EXEC" in white capital letters.

Rules for Coding

1. Code an EXEC statement for each job step.
2. 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 1 through 8 alphanumeric and national (@,#,\$) characters. The first character must be an alphabetic or national character.

3. The two types of parameters that can be coded in the operand field of the EXEC statement are:

Positional parameters, which 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, which 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  
DPRTY  
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
```

The 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.

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

EXEC

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 consist of up to 8 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        DSNAME=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        DSNAME=%%PARTDS(PROG),UNIT=2314,DISP=(MOD,PASS), X
              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 $\epsilon\epsilon$ PARTDS and will be executed in the step named EXECUTE. $\epsilon\epsilon$ PARTDS will be deleted at the end of the step named EXECUTE.

```
//STEP2      EXEC      PGM=UPDT
//DDA        DD         DSN=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 name STEP3.

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

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

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 in a job you want to assemble, linkage edit, and then execute a program, you must make the output of the linkage editor a member of a partitioned data set. This is accomplished by creating a temporary library. A temporary library is a partitioned data set created in the job to store a program as a member of the data set, until it is executed in a subsequent job step. When the program is required, you may refer back to the DD statement that defines the temporary library and the member by coding PGM=*.stepname.ddname or PGM=*.stepname.procstepname.ddname. You may also request use of a program that is a member of a temporary library by coding PGM=program name and including a DD statement named JOBLIB or STEPLIB that defines the temporary library.

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

System Library

The system library is a partitioned data set named SYS1.LINKLIB and it contains frequently used programs, as well as system programs. You 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 may also be executed by coding PGM=*.stepname.ddname or PGM=*.stepname.procstepname.ddname. This can be done only when the named DD statement defines the program as a member of the system library.

Private Library

A private library is a partitioned data set that contains programs not used frequently enough to warrant their inclusion in the system library. You request use of a program that is a member of a private library by coding `PGM=program name` and including a DD statement named `JOBLIB` or `STEPLIB` that defines the private library. The system automatically looks in the private library and, if the program is not found there, in `SYS1.LINKLIB` for a member with the corresponding name.

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

A black circular logo with the word "EXEC" in white capital letters.

The IEFBR14 Program

If space allocation and disposition processing requests are contained in your job control statements, you can satisfy these requests prior to 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.)

The PROC Parameter

Keyword, 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 Appendix B of this publication.

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

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 will be 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 PAYWKRS is to be called. This specification has the same effect as coding PROC=OPERATE.

The 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/VS1 Planning and Use Guide, GC24-5090** or **OS/VS2 Planning and Use Guide, GC28-0600**.

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

accounting information

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

EXEC

Rules for Coding

1. If the accounting information consists of only one subparameter, you need not code 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 will then pertain 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. Since 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,          X  
//          ACCT.BILL='121+366'
```

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

The ADDRSPC Parameter

Keyword, Optional

The ADDRSPC parameter can be used to assure 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/VS JCL Services, GC28-0617.

$$\text{ADDRSPC} = \begin{cases} \text{VIRT} \\ \text{REAL} \end{cases}$$

Rules for Coding

1. For OS/VS1, code the REGION parameter to specify how much real storage you need.
For OS/VS2, code the REGION parameter to specify how much real or virtual 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 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 will be ignored and the value on the JOB statement will be used.
4. For OS/VS1, when you specify ADDRSPC=REAL, the REGION size you specify overrides the system partition size. Therefore, make the REGION size large enough to satisfy all storage requests for a job step, keeping in mind that the REGION size cannot be larger than the partition size.

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=VIRTUAL
```

The ADDRSPC parameter requests virtual storage. Since the REGION parameter is omitted, the default region size for the installation will be used.

The COND Parameter

Keyword, Optional

The COND parameter specifies whether a job step will be executed based on completion 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/VS JCL Services*, GC28-0617.

$$\text{COND}=(\left[\begin{array}{l} (\text{code}, \text{operator}) \\ (\text{code}, \text{operator}, \text{stepname}) \\ (\text{code}, \text{operator}, \text{stepname}.\text{procstepname}) \end{array} \right] , \dots [,] \left[\begin{array}{l} \text{EVEN} \\ \text{ONLY} \end{array} \right])$$

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 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 will be 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 if any tests are satisfied, this job step will be bypassed.

Rules for Coding

1. If you code neither EVEN nor 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 need not code outer parentheses.

EXEC

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) will 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.

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 will bypass this step. (A return code of 4 or greater allows this step to be executed.) Since neither EVEN nor ONLY is specified, this job step will be automatically bypassed if a preceding step abnormally terminates.

```
//TEST2      EXEC          PGM=BACK,COND=(( 16,GE ),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 will be 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 will be 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 will be bypassed. Since neither EVEN nor ONLY is specified, this job step will be automatically bypassed if a preceding step abnormally terminated.

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

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.



The DPRTY Parameter (For OS/VS2)

Keyword, Optional

The DPRTY parameter assigns a dispatching priority to a job step. Dispatching priority determines the order in which tasks will use real storage.

For further information on use of the DPRTY parameter, see *OS/VS JCL Services*, GC28-0617.

DPRTY=(value1,value2)

value1

a number from 0 through 15 which indicates whether the job step is to have the same priority or a different priority than the job. The job priority is coded in the PRTY parameter on the JOB statement.

value2

a number from 0 through 15 which the system adds to the internal priority to form the dispatching priority. The system forms the internal priority by converting the value assigned to value1 in the DPRTY parameter.

Rules for Coding

1. Do not assign a number of 15 to value1. This number is used for system tasks.
2. If you omit value1, you must code a comma preceding value2 to indicate the absence of value1.
3. If you omit value2, you need not code the parentheses.
4. If the DPRTY parameter is coded for OS/VS1, the parameter is checked for syntax but otherwise ignored.
5. **Defaults:** If you do not code the DPRTY parameter, the job step is assigned the priority assigned to the job either on the JOB statement (the PRTY parameter) or by default.
If you do not assign a number to value1, 0 is assumed.
If you do not assign a number to value2, 11 is assumed.
6. When you want the job step to have a different dispatching priority than the job, code the DPRTY parameter and either raise or lower the values, depending on whether the step is to have a higher or lower priority than the job.
7. You can code the DPRTY parameter on the EXEC statement of a cataloged procedure.
8. If a job step calls a cataloged procedure:

To override all DPRTY parameters, in the cataloged procedure code the DPRTY parameter on the EXEC statement that calls the procedure. This will establish one dispatching priority for all steps in the procedure.

To override only certain DPRTY parameters, code, on the EXEC statement that calls the procedure, DPRTY.procstepname for each procedure step that you want to override. The dispatching priority will then pertain only to the named procedure step.

Examples of the DPRTY Parameter

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

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

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

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

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

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

EXEC

The 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/VS Supervisor Services and Macro Instructions*, GC27-6979.

PARM=value

value

up to 100 characters of information which the system is to pass 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, e.g., PARM='P1,123,MT5' or PARM=(P1,123,MT5).

Enclosing apostrophes and parentheses are not passed to the processing program; commas within apostrophes and parentheses are passed as part of the value.

2. If 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, e.g., PARM='P50,12+80' or PARM=(P50,'12+80'). (The enclosing apostrophes and parentheses are not considered part of the value.)

If one of the special characters is an ampersand and you are not defining a symbolic parameter, code two consecutive ampersands in its place, e.g., PARM='3462&&5'.

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

3. If 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 is passed to the first procedure step and PARM parameters in all other procedure steps are nullified.

To override some of the PARM parameters contained in the procedure, code, on the EXEC statement that calls the procedure, PARM.procstepname for each procedure step that you want to override. Information provided in the PARM value will be passed only to the named procedure step.

Example of the PARM Parameter

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

Passes the information in the PARM parameter, except the apostrophes, 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 procedure steps in LKED contain the PARM parameter these parameters will still be effective.

A black circular logo with the word "EXEC" in white, sans-serif capital letters.

The 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, refer to:

- OS/VS Checkpoint/Restart, GC26-3784.

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

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 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. For OS/VS1 when RD=R or RD=RNC is specified, you must code MSGLEVEL=(1,0), MSGLEVEL=(1,1), or MSGLEVEL=1 on the JOB statement. For OS/VS2, you may omit the MSGLEVEL parameter.
2. If automatic step restart is permitted, i.e., RD=R or RD=RNC is coded, assign each job step a unique step name.
3. Code the RD parameter on EXEC statements instead of the JOB statement when you want to make different restart requests for each job step.

If 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 can be coded on the EXEC statement of a cataloged procedure. If the job step does call 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 will then pertain only to the named procedure step.

EXEC

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; 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).

The REGION Parameter

Keyword, Optional

For OS/VS1, the REGION parameter specifies the amount of real storage to be allocated to a job step.

For OS/VS2, the REGION parameter specifies how much real or virtual storage is to be allocated to a job step. The REGION parameter should be specified or the REGION size default value in the input reader procedure will be used.

For further information on the REGION parameter, see *OS/VS JCL Services*, GC28-0617.

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 is the input reader procedure will be used.
3. If you code the REGION parameter on the JOB statement, REGION parameters on the job's EXEC statements will be ignored.
4. In OS/VS1, if you code ADDRSPC=VIRTUAL with the REGION parameter, REGION will be ignored.
5. In OS/VS1, the REGION size you specify overrides the system partition size. Therefore, make the REGION size large enough to satisfy all real storage requests for a job step, keeping in mind that the REGION size must be larger than the partition size.

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.

The 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/VS System Management Facilities (SMF), GC35-0004.

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

EXEC

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.

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 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. Code the TIME parameter on the JOB statement if you want to indicate how long the entire job can use the CPU.
5. 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.

6. **Default:** When you do not code the TIME parameter on either the JOB statement or the EXEC statement, a default time limit, specified by the installation, is assumed for each job step.
7. The remaining job time may affect the amount of time the step can use the CPU. If the remaining CPU time for the job is less than the CPU time limit specified on the EXEC statement, the step can use the CPU only for the job's remaining CPU time.

8. **System Management Facilities:** Normally, a step that exceeds the specified time limit causes termination of the job. However, if the System Management Facilities are in use in the system, a user exit routine can extend the time limit so that processing can continue.

Without the System Management Facilities 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 causes termination of the job.

With the System Management Facilities in use, the installation determines the time limit for wait states. A job step remaining in a wait state for more than the established time limit causes termination of the job unless a user-provided exit routine extends the wait state limit for that step.

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 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.

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 -- the 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 1 through 8 alphameric 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), there are two instances when you should not code a ddname at all:
 - 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 wish to make use of the particular facilities which these names represent to the system; these facilities are explained in detail in the following pages.

```
JOBLIB  
STEPLIB  
SYSABEND  
SYSUDUMP  
SYSCHK
```

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

```
*  
DATA  
DUMMY  
DYNAM
```

The keyword parameters are:

AFF	DLM	QNAME	TERM (for OS/VS1)
COPIES	DSNAME	SEP	TERM (for OS/VS2)
DCB	FCB	SPACE	UCS
DDNAME	HOLD	SPLIT	UNIT
DEST	LABEL	SUBALLOC	VOLUME
DISP	OUTLIM	SYSOUT	

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, followed by the name of the DD statement that you want to override.
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, followed by a ddname of your choosing. Added statements must follow all overriding statements.
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 will override parameters on the statement named DAY. If the step does not include a DD statement named DAY, this statement will be 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.

The JOBLIB Facility

DD statement

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

General Rules for Coding

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 first searches the private library; if it does not find the program there, it next 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 first 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. One of the following combinations of DISP parameter values must be coded:

```
DISP=(NEW,PASS)
DISP=(OLD,PASS)
DISP=(SHR,PASS)
DISP=(NEW,CATLG)
DISP=(OLD,CATLG)
DISP=(SHR,CATLG)
```

For further explanation, refer to the DISP parameter in the index of this publication.

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

Rules for Coding When the Library Is Cataloged

1. Code the DSNNAME parameter to specify the name of the private library.
2. Code the DISP parameter. The DISP parameter must be one of the following values:

```
DISP=(OLD,PASS)
DISP=(OLD,KEEP)
DISP=(OLD,CATLG)
```
3. Code the DCB parameter if complete data control block information is not contained in the data set label.
4. 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.

Rules for Coding When the Library Is Not Cataloged

1. Code the DISP parameter. The DISP parameter must be one of the following values:

```
DISP=(OLD,PASS)
DISP=(SHR,PASS)
DISP=(NEW,PASS)
```

2. Code the UNIT parameter to specify the device to be allocated to the library.
3. Code the DSNNAME parameter unless the data set has been assigned a disposition of (NEW,PASS).
4. Code the VOLUME parameter unless the status of the data set is NEW.
5. 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.
6. Code the DCB parameter if complete data control block information is not contained in the data set label.
7. To refer to the private library in a later DD statement, code
DSNNAME=*.JOB LIB,VOLUME=REF=*.JOB LIB (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=*.JOB LIB to obtain volume and unit information.

Examples of the JOBLIB DD Statement

```
//PAYROLL      JOB
//JOBLIB       DD      DSNNAME=PRIVATE.LIB4,DISP=(OLD,PASS)
//STEP1       EXEC    PGM=SCAN
//STEP2       EXEC    PGM=UPDATE
//DD1         DD      DSNNAME=*.JOB LIB,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      DSNNAME=PRIV.DETP58,DISP=(OLD,SHR),          X
//              UNIT=2314,VOLUME=SER=D58PVL
//STEP         EXEC    PGM=DAY
//STEP2       EXEC    PGM=BENEFITS
//DD1         DD      DSNNAME=*.JOB LIB,VOLUME=REF=*.JOB LIB,      X
//              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          EXEC    MSGLEVEL=( 1, 1 )
//JOBLIB        DD      DSNAMES=GROUP8.LEVEL5,DISP=( NEW,CATLG ),      X
//              DD      UNIT=2314,VOLUME=SER=148562,                  X
//              DD      SPACE=( CYL,( 50,3,4 ) )                      X
//STEP1         EXEC    PGM=DISC
//DDA           DD      DSNAMES=GROUP8.LEVEL5(RATE),DISP=OLD,        X
//              DD      VOL=REF=* .JOBLIB
//STEP2         EXEC    PGM=RATE

```

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

```

//PAYROLL      JOB
//JOBLIB        DD      DSNAMES=KRG.LIB12,DISP=( OLD,PASS )
//              DD      DSNAMES=GROUP31.TEST,DISP=( OLD,PASS )
//              DD      DSNAMES=PGMSLIB,UNIT=2314                    X
//              DD      DISP=( OLD,PASS ),VOLUME=SER=34568

```

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



The STEPLIB Facility

DD Statement

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

General Rules for Coding

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 next 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 a cataloged procedure.
6. To concatenate libraries, i.e., to arrange a sequence of DD statements that define different data sets:

Code STEPLIB as the ddname on the first DD statement.

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 DSN=SYS1.LINKLIB,DISP=OLD
```

For the particular job step, the system will first search 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 below under separate headings.

Rules for Coding When the Library Is Cataloged

1. Code the DSN 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 data control block information is not contained in the data set label.

Rules for Coding When the Library Has Been Passed By a Previous Step

1. To be available for use by subsequent job steps within a job, a previously defined step library must have an assigned disposition of PASS.

2. To reference a previously defined step library:

Code the DSNNAME 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, i.e., `*.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 data control block information is not contained in the data set label.

DD

Rules for Coding When the Library Is Neither Cataloged Nor Passed

1. Code the DSNNAME 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 data control block 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 DSNNAME=PRIV.LIB5,DISP=(OLD,KEEP)
//STEP3 EXEC PGM=TIL80
//STEPLIB DD DSNNAME=PRIV.LIB13,DISP=(OLD,KEEP)
```

The private libraries defined in STEP2 and STEP3 are cataloged.

```
//PAYROLL JOB
//JOBLIB DD DSNNAME=LIB5.GROUP4,DISP=(OLD,PASS)
//STEP1 EXEC PROC=SNZ12
//STEP2 EXEC PGM=SNAP10
//STEPLIB DD DSNNAME=LIBRARYP,DISP=(OLD,PASS), X
// UNIT=2314,VOLUME=SER=55566
//STEP3 EXEC PGM=A1530
//STEP4 EXEC PGM=SNAP11
//STEPLIB DD DSNNAME=*.STEP2.STEPLIB, X
// DISP=(OLD,KEEP)
```

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

```
//PAYROLL  JOB
//JOBLIB   DD          DSNAME=LIB5.GROUP4,DISP=(OLD,PASS)
//STEP1    EXEC        PGM=SUM
//STEPLIB  DD          DSNAME=SYS1.LINKLIB,DISP=OLD
//STEP2    EXEC        PGM=VARY
//STEP3    EXEC        PGM=CALC
//STEPLIB  DD          DSNAME=PRIV.WORK,DISP=(OLD,PASS)
//         DD          DSNAME=LIBRARYA,DISP=(OLD,KEEP),      X
//         UNIT=2314,VOLUME=SER=44455
//         DD          DSNAME=LIB.DEPT88,DISP=(OLD,KEEP)
//STEP4    EXEC        PGM=SHORE
```

All steps can use programs contained in the private library named LIB5.FEOUP4, which is defined in the JOBLIB DD statement. STEP1 can use a program from the system library, since 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, LIB5.GROUP4, SYS2.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.

The 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 abnormally terminates. 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 abnormally terminates. The dump provided by this facility includes only the processing program storage area.

For information on how to interpret dumps, see *OS/VS1 Debugging Guide*, GC24-5093 or *OS/VS2 Debugging Guide*, GC28-0632.

DD

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 specify a specific direct access device on which the dump is to be stored, code the UNIT parameter. Otherwise, the system will assign 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 the data set's status and disposition. Since you want to store the data set, make the data set's conditional disposition KEEP or CATLG.

4. If you want to store the dump on a direct access device, code wither the SPACE, SPLIT, or SUBALLOC parameter, specifying the amount of space you want allocated to the data set.

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        DDSNAME=DUMP,UNIT=2314,DISP=(,PASS,KEEP),    X
//          //        VOLUME=SER=1234
//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 that the dumping operation will not be inhibited due to insufficient space; if STEP1 does not abnormally terminate but STEP2 does, the dump will be written using the space allocated in STEP1. In both steps, a conditional disposition of KEEP is specified. This will allow storing of the dump if either of the steps abnormally terminates. If both of the steps are successfully executed, the second subparameter of the DISP parameter (DELETE) in STEP2 will cause 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,        X
//          //        KEEP),VOLUME=SER=54366
//STEP2      EXEC      PGM=PRINT,COND=ONLY
//IN         DD        DNAME=*.STEP1.SYSUDUMP,DISP=(OLD,DELETE),    X
//          //        VOLUME=REF=*.STEP1.SYSUDUMP
```

Step 1 specifies that the dump is to be stored if the step abnormally terminates. Because COND=ONLY is specified in STEP2, the step will be executed only if STEP1 abnormally terminates. STEP2 uses a program that prints the dump.

The 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 the following publication:

OS/VS Checkpoint/Restart, GC26-3784

General Rules for Coding

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, i.e., 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 will be ignored.
5. The rules for coding parameters on the SYSCHK DD statement depend on whether or not the checkpoint data set is cataloged. These rules are discussed below under separate headings.

When the Checkpoint Data Set is Cataloged

1. Code the DSNAME 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 non-standard labels or no labels, code DCB=TRTCH=C.

When the Checkpoint Data Set is not Cataloged

1. Code the DSNAME parameter, specifying the name of the checkpoint data set. If the checkpoint data set is partitioned, do not include a member name in the DSNAME 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 on 7-track magnetic tape with non-standard or no labels, code DCB=TRTCH=C.

Examples of the SYSCHK DD Statement

```
//JOB1      JOB      RESTART=( STEP3,CK3 )
//SYSCHK    DD      DSNAME=CHLIB,UNIT=2314,          X
//          DD      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      DSNAME=PRIV.LIB3,DISP=( OLD,PASS )
//SYSCHK    DD      DSNAME=CHECKPTS,DISP=( OLD,KEEP ),  X
//          DD      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      DSNAME=CHKPTLIB,DISP=OLD,          X
//          DD      LABEL=( ,NSL ),DCB=TRTCH=C
//STEP1     EXEC
```

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

The * Parameter

Optional

* the parameter specifies that for a processing program data follows the DD statement. The * parameter causes the system to check 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. Code only the DCB subparameters BLKSIZE, BUFNO, and DLM on a DD * statement. Any other parameters coded on a DD * statement will cause job failure.
6. Code the DATA parameter instead of the * parameter when the data contains statements starting with //.
7. When you precede the data with a DD * statement, a delimiter statement (/*) following the data is optional.
8. You may include input stream data on any device supported by QSAM, e.g., a card reader, a magnetic tape, or a direct access device.
9. You must code input stream data records in BCD or EBCDIC.
10. **Separating groups of data:** You can include several distinct groups of data in the input stream for a job step or a procedure step. The system will recognize 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 for a group of data, the system provides a DD * statement having SYSIN as its ddname.)
11. If the processing program does not read all the data in an input stream, the remaining data is flushed without causing abnormal termination of the job.
12. When a job is submitted via remote job entry and the DCB subparameter BUFNO is coded on a DD * statement, BUFNO is ignored.
13. **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 ),DIAGNS=TRACE
                  .
                  data
                  .
/*

```

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

```

//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.

```

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

```

Defines data in the input stream. These DCB subparameters will override those specified in the input reader procedure.

The 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. Code only the DCB subparameters BLKSIZE, BUFNO, and DLM on a DD DATA statement. Any other parameters coded on a DD DATA statement will cause job failure.
5. Code the * parameter instead of the DATA parameter when the data does not contain statements starting with //.
6. **Separating groups of data:** 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.
7. You can place input stream data on any device supported by QSAM, e.g., a card reader, a magnetic tape, or a direct access device.
8. You must code input stream data records in BCD or EBCDIC.
9. If the processing program does not read all the data in an input stream, the remaining data is flushed without causing abnormal termination of the job.
10. When a job is submitted via remote job entry and the DCB subparameter BUFNO is coded on a DD statement, BUFNO is ignored.
11. **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      DSNAME=A.B.C,VOLUME=SER=D88,          X
//              DD      UNIT=2314,SPACE=(TRK,(10,5)),          X
//              DD      DISP=(,CATLG,DELETE)
//PREP.INPUT     DD      DATA
                .
                .
                data
                .
                .
/*
//ADD.DD6        DD      *
//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. Since 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,DIAGNS=TRACE
                .
                .
                data
                .
                .
/*
```

Defines several 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. The DCB subparameter DIAGNS requests the OPEN/CLOSE/EOV trace option.

The 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/VS JCL Services*, GC28-0617.

```
//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 DSNNAME parameter on the overriding DD statement. However, be sure that the data set name is not NULLFILE. Assigning the name NULLFILE in the DSNNAME parameter has the same effect as coding DUMMY.
4. If you code the DUMMY parameter and also request an access method other than the basic sequential access method (BSAM) or queued sequential access method (QSAM) to read or write the data set, a programming error will occur.
5. Besides bypassing input or output operations on a data set, the DUMMY parameter causes the UNIT, VOLUME, SPACE, and DISP parameters, when coded on the DD DUMMY statement, to be ignored; however, these parameters are checked for syntax.
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 will be assigned a dummy status.

Examples of the DUMMY Parameter

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

This DD statement defines a dummy data set. The parameter 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 will supply information that was not supplied in the DCB macro instruction for the data control block. Otherwise, abnormal termination 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
```

If you are calling a cataloged procedure that contains the following DD statement in a procedure step named LOCK,

```
//MSGSDD SYSOUT=A
```

you can make this DD statement define a dummy data set by coding:

```
//LOCK.MSGSDD DUMMY
```

The DYNAM Parameter (For OS/VS2 with TSO)

Positional, Optional

The DYNAM parameter is coded in the TSO LOGON procedure to specify that dynamic allocation of data sets is to be used. This allows a TSO user to defer definition of a data set until it is required. During LOGON processing for TSO, no devices or external storage are allocated to a data set defined by a DD DYNAM statement. The DYNAM parameter reserves space in internal tables so that data set requirements that arise during a terminal session can be satisfied. When a user requires a data set, the actual device and external storage for the data set can then be allocated.

For further information on the use of the DYNAM parameter, see *OS/VS2 TSO Guide*, GC28-0644.

```
//ddname          DD      DYNAM
```

DD

Rules for Coding

1. Do not code any other parameters with the DYNAM parameter.
2. Do not use the DDNAME parameter to refer to a DYNAM DD statement.
3. For OS/VS1, DYNAM has the same meaning as the DUMMY parameter.
4. When you use DYNAM in the background (batch environment) or in the foreground before allocation, it has the same effect as coding DUMMY. Refer to the section on the DUMMY parameter in this book for more information.
5. To nullify the DYNAM parameter in a cataloged procedure, code the SYSOUT or DSNAME parameter in the overriding DD statement, but do not use the DSNAME "NULLFILE".

Example of the DYNAM Parameter

```
//INPUT          DD      DYNAM
```

For TSO, this statement specifies dynamic allocation. For background jobs, this statement has the same meaning as a DUMMY DD statement.

The 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.
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 "The 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 will decide that, based on what devices are available for allocation.
5. **Satisfying 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 operation intervention. The operator is given 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. **Non-specific request:** If you make a non-specific 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 would permit 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        DSNAME=A.B.C,DISP=OLD
//INPUT2     DD        DSNAME=FILE,DISP=OLD,UNIT=2400,           X
//          VOLUME=SER=54333
//BUF        DD        UNIT=2400,SEP=( INPUT1,INPUT2 )
//OUTPUT     DD        DSNAME=ALPHA,UNIT=TAPE,DISP=( ,KEEP ),AFF=BUF
```

The system attempts to assign the data sets defined by the DD statements BUF and OUTPUT to a channel other than the ones assigned to the data sets defined by the DD statements INPUT1 and INPUT2. The data sets defined by the DD statements BUF and OUTPUT may or may not be assigned to the same channel. The parameter SEP=(INPUT1,INPUT2) could have been coded instead of AFF=BUF.

DD

The COPIES Parameter

Keyword, Optional

The COPIES parameter allows you to request more than one copy of the output data sets. For further information of the use of the COPIES parameter, refer to *OS/VS JCL Services*, GC28-0617.

```
COPIES=nnn
```

nnn

the number of copies of the SYSOUT data set to be written to the printer, punch, or tape.

Rules for Coding

1. You can code the COPIES parameter only with the SYSOUT parameter on a DD statement.
2. **Default:** If you do not code the COPIES parameter 1 is assumed.
3. The maximum number of copies you can specify is 255, the minimum is one.

Examples of the COPIES Parameter

```
//RECORD DD SYSOUT=W,COPIES=32
```

Requests 32 copies of the data set record.

The DCB Parameter

Keyword, Optional

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

For further information on the formation of the data control block, see *OS/VS Data Management Services Guide, GC26-3783*.

```
      { DCB=      (list of attributes)
        DCB=      ( dsname
                    { *.ddname
                      *.stepname.ddname
                      *.stepname.procstepname.ddname }
                  ) [,list of attributes] ) }
```

list of attributes

those DCB keyword subparameters that describe the data set and are needed to complete the data control block. The DCB keyword subparameters are listed in this section in the immediately following pages. They are arranged alphabetically, according to access methods, e.g., 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).

General Rules for Coding

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 data control block is completed by another source, e.g., the DCB macro instruction in the processing program. There are several ways of specifying DCB information on the DD statement. 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.

Supplying 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 data control block 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 in this section in the immediately following pages. They are arranged alphabetically, according to access methods, e.g., BDAM, BISAM.

Copying 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 which is opened for input. A permanently resident volume is the most likely 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. The following 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 and expiration date of the cataloged data set will also be 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 in this section in the immediately following pages. They are arranged alphabetically, according to access methods, e.g., BDAM, BISAM.

Copying 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 in this section in the immediately following pages. They are arranged alphabetically, according to access methods, e.g., BDAM, BISAM.

DD

Examples of the DCB Parameter

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

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

```
//DD2 DD DSNAME=BAL,DISP=OLD,DCB=(RECFM=F,LRECL=80, X  
// BLKSIZE=80)  
//DD3 DD DSNAME=CNANN,DISP=(,CATLG,DELETE),UNIT=2400, X  
// LABEL=(,NL),VOLUME=SER=663488,DCB=*.DD2
```

The statement named DD3 defines a new data set and requests the system to copy the DCB subparameters from the DD statement named DD2, which is in the same job step.

```
//DD4 DD DSNAME=JST,DISP=(NEW,KEEP),UNIT=2314, X  
// SPACE=(CYL,(12,2)),DCB=(A.B.C,KEYLEN=8)
```

This DD statement defines a new data set and requests the system to copy DCB information from the data set label of the cataloged data set named A.B.C. If the data set label contains a key length specification, it is overridden since KEYLEN is coded on the DD statement.

```
//DD5 DD DSNAME=SAME,DISP=OLD,UNIT=2314, X  
// DCB=(*.STEP1.PROCSTP5.DD8,BUFNO=5)
```

This DD statement defines an existing data set and requests the system to copy the DCB subparameters from the DD statement named DD8, which is contained in the procedure step named PROCSTP5. The cataloged procedure was called by the job step named STEP1. If any of the DCB subparameters coded on the procedure DD statement have been previously defined for this data set, they are ignored. If the BUFNO subparameter has not been previously specified for the data set, then five buffers are assigned to the data control block.

The DCB Subparameters for BDAM

You can use the following DCB subparameters with BDAM:

BFALN	DIAGNS	OPTCD
BLKSIZE	DSORG	RECFM
BUFL	KEYLEN	
BUFNO	LIMCT	

These subparameters are defined and the rules for coding them are explained below.

The 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 full-word 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.

The BLKSIZE Subparameter

The BLKSIZE subparameter specifies the maximum length, in bytes, of a block. The largest number allowed is 32,760 except for blocks of ASCII records on magnetic tape, this maximum length is 2048 and the minimum length is 18.

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

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

- If RECFM=F, then BLKSIZE must be logical record length.
- If RECFM=FB, then BLKSIZE must be an integral multiple of the logical record length.
- If RECFM=V, then BLKSIZE must be (maximum block size + 4).
- If RECFM=VB, then BLKSIZE must be (n X 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 block size specified in the label.

The BUFL Subparameter

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

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

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



The BUFNO Subparameter

The BUFNO subparameter specifies how many buffers are to be assigned to the data control block; 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 as follows:

Method of obtaining the buffer pool:

BUILD macro instruction
GETPOOL macro instruction

Dynamic buffering
Automatically.

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.
Must be specified.

The 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 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 Generalized Trace Facility are MODE=EXT and TRACE=USR.

The 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 access

DAU
direct access, unmovable

PS
physical sequential

PSU
physical sequential, unmovable

The DSORG subparameter must be coded on the DD statement that defines the data set. When creating the data set, the DSORG subparameter must be coded as DA or DAU on the DD statement that defines the data set and PS or PSU in the DCB macro instruction.

The 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.

The 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.

The OPTCD Subparameter

The OPTCD subparameter requests optional services from the control program.

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

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 the following 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.

The 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.

$$\text{RECFM} = \left\{ \begin{array}{ll} \text{U} & \text{VS} \\ \text{V} & \text{VBS} \\ \text{F} & [\text{T}] \end{array} \right\}$$

The following 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.

F

indicates that the records are of fixed length.

T

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

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

The DCB Subparameters for BISAM

You can use the following DCB subparameters with BISAM:

BFALN BUFNO NCP
BUFL DIAGNS

These subparameters are defined and the rules for coding them are explained below.

The 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 full-word boundary that is not also a doubleword boundary.

D

indicates that each buffer starts on a doubleword.

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

The BUFL Subparameter

The BUFL parameter 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 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.

The BUFNO Subparameter

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

$$\text{BUFNO} = \text{number of buffers}$$

Requirements for coding the BUFNO subparameter are as follows:

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.



The 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 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 Generalized Trace Facility are MODE=EXT and TRACE=USR.

The 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 value with the DSORG subparameter:

IS
 indexed sequential

The DSORG subparameter must be coded on the DD statement.

The 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, 1 is assumed.

The maximum number is normally 99, but may be less, based on limits established when the system was generated.

If chained scheduling is used, NCP must be specified as more than 1.

The DCB Subparameters for BPAM

You can use the following DCB subparameters with BPAM:

BFALN	DIAGNS	OPTCD
BLKSIZE	KEYLEN	RECFM
BUFL	LRECL	
BUFNO	NCP	

These subparameters are defined and the rules for coding them are explained below.

The 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 full-word boundary that is not also a doubleword boundary.

D
indicates that each buffer starts on a doubleword.

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

The BLKSIZE Subparameter

The BLKSIZE subparameter specifies the maximum length, in bytes, of a block. The maximum length 32,760. For blocks of ASCII records on magnetic tape, the largest number that can be specified is 2048 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, then BLKSIZE must be logical record length.
- If RECFM=FB, then BLKSIZE must be an integral multiple of the logical record length.
- If RECFM=V, then BLKSIZE must be (maximum block size + 4).
- If RECFM=VB, then BLKSIZE must be (n X 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 block size specified in the label.

The 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 block size and key length information is used to establish the buffer length.



The BUFNO Subparameter

The BUFNO subparameter specifies the number of buffers to be assigned to the data control block; the maximum normally 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 as follows:

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.

The 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 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 Generalized Trace Facility are MODE=EXT and TRACE=USR.

The 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 values with the DSORG subparameter:

- PO
partitioned organization.
- POU
partitioned organization, unmovable

The 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

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.

The 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.

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

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

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



The 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, 1 is assumed.

The largest number that can be specified in the NCP subparameter is 99 but may be less, based on limits established when the system was generated.

If chained scheduling is used, NCP must be specified as more than 1.

The OPTCD Subparameter

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

OPTCD= $\left\{ \begin{array}{c} C \\ W \\ WC \end{array} \right\}$

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.

The following values can be used with the OPTCD subparameter:

C

requests that chained scheduling be used. Chained scheduling will be ignored unless ADDRSPC=REAL has been specified.

W

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

The 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.

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

The following 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.

The DCB Subparameters for BSAM

You can use the following DCB subparameters with BSAM:

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

These subparameters are defined and the rules for coding them are explained below.

The 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 full-word boundary that is not also a doubleword boundary.

D

indicates that each buffer starts on a doubleword.

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

The BLKSIZE Subparameter

The BLKSIZE subparameter specifies the maximum length, in bytes, of a block. The largest number that can be specified in the BLKSIZE subparameter is 32,760 except for blocks of ASCII records on magnetic tape, the number is 2048 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, then BLKSIZE must be logical record length.
- If RECFM=FB, then BLKSIZE must be an integral multiple of the logical record length.
- If RECFM=V, then BLKSIZE must be (maximum block size + 4).
- If RECFM=VB, then BLKSIZE must be (n X logical record length) + 4, where n is the number of logical records in the block.
- If RECFM=D or RECFM=DB, then 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 block size specified in the label.

DD

BSAM

The BUFL Subparameter

The BUFL subparameter specifies the length, in bytes, of each buffer in the buffer pool. 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 block size and key length information is used to establish buffer length.

The BUFNO Subparameter

The BUFNO subparameter specifies how many buffers are to be assigned to the data control block; the maximum normally 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 as follows:

Method of obtaining the buffer pool:

BUILD macro instruction
GETPOOL macro instruction
Automatically.

Requirement for indicating the number of buffers:

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

The 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.

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 29. 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.

The 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 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 you omit the CODE subparameter, I is assumed.



The DEN Subparameter

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

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

- 0 for 7 track tape, indicates 200 bits per inch.
- 1 for 7 track tape, indicates 556 bits per inch.
- 2 for 7 track tape, indicates 800 bits per inch; for 9 track tape, indicates 800 bits per inch.
- 3 for 9 track tape, indicates 1600 bits per inch.

Default: If you omit the DEN subparameter:

800 bits per inch is assumed for 7 track tape and for 9 track tape without dual density.

1600 bits per inch is assumed for 9 track tape with dual density or phase-encoded drives.

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

Do not specify DEN for a SYSOUT data set.

The 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 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 Generalized Trace Facility are MODE=EXT and TRACE=USR.

The 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 values with the DSORG subparameter:

PS

physical sequential

PSU

physical sequential, unmovable

The FUNC Subparameter

The FUNC subparameter specifies the type of data set to be opened for the 3525 Card-Read-Punch-Print.

FUNC=function

You can code the following functions with the FUNC subparameter:

I

interpret-punch data set

R

read

P

punch

W

print

D

data protection for a punch data set

X

printer

T

two-line printer

The only valid combinations of these values are:

I	WT	RWX	PWXT
R	WXT	RWT	RPW
P	RP	RWXT	RPWX
W	RPD	PW	RPWXT
WX	RW	PWX	RPWD

Default: If the function information is not supplied by any source, P is assumed.

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

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

The record length can be omitted from all sources, in which case the block size specification (BLKSIZE) is used.

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

For ASCII records on magnetic tape, the maximum record length is 2048 bytes and the minimum record length is 18 bytes.

DD

BSAM

The MODE Subparameter

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

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

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, and TRTCH are mutually exclusive. Therefore, do not code any of these other subparameters with MODE.

The 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, 1 is assumed.

The largest number that can be specified in the NCP subparameter is 99, based on limits established when the system was generated.

If chained scheduling is used, NCP must be specified as more than 1.

The OPTCD Subparameter

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

OPTCD= {
B
C
H
Q
T
U
W
Z
UC
WC
ZC

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.

The following values can be used with the OPTCD subparameter:

- B**
requests that end-of-file recognition be disregarded for tapes.
- C**
requests that chained scheduling be used. Chained scheduling will be ignored unless ADDRSPC=REAL has been specified.
- H**
requests hopper empty exit for Optical Readers.
- 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**
only for 1403 printers with the Universal Character Set feature; unblocks data checks and allows analysis by an appropriate error analysis routine.
- If U is omitted, data checks are blocked, i.e., not recognized as errors.
- 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.

The PRTSP Subparameter

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

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

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.

The 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.

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

For BSAM using ASCII data sets on tape:

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

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

The following 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.

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, i.e., no truncated blocks or unfilled tracks within the data set, with the exception of the last block or track.
for variable-length records, a record may span more than one block.
- 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. Variable length records cannot be in ASCII.



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

The STACK Subparameter

The STACK subparameter specifies which stacker bin is to receive 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.

The TRTCH Subparameter

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

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

- 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 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 is assumed.

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

The DCB Subparameters for BTAM

You can use the following DCB subparameters with BTAM:

BFTEK EROPT
BUFNO
DIAGNS
DSORG

These subparameters are defined and the rules for coding them are explained below.

The 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.

The BUFNO Subparameter

The BUFNO subparameter specifies the number of buffers to be assigned to the data control block; 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 as follows:

Method of obtaining the buffer pool:

BUILD macro instruction
GETPOOL macro instruction

Automatically

Requirement for indicating the number of buffers:

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

The 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 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 Generalized Trace Facility are MODE=EXT and TRACE=USR.

DD

BTAM

The 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.

The EROPT Subparameter

The EROPT subparameter requests the BTAM on-line terminal test option.

EROPT=T

T

requests the BTAM on-line terminal test option.

The DCB Subparameters for EXCP

You can use the following DCB subparameters with EXCP:

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

These subparameters are defined and the rules for coding them are explained below.

The 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 full-word boundary that is not also doubleword boundary.

D indicates that each buffer starts on a doubleword.

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

The BFTEK Subparameter

The BFTEK subparameter specifies the type of buffering to be 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.

The 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.

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

The BUFNO Subparameter

The BUFNO subparameter specifies the number of buffers to be assigned to the data control block; the maximum that can be specified in the BUFNO subparameter is 255, but the actual number allowed may be less than 255 because of limits established when the system was generated.

$$\text{BUFNO} = \text{number of buffers}$$


Requirements for coding the BUFNO subparameter are as follows:

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.

The 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.

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

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.

The DEN Subparameter

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

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

0

for 7 track tape, indicates 200 bits per inch.

1
for 7 track tape, indicates 556 bits per inch.

2
for 7 track tape, indicates 800 bits per inch; for 9 track tape, indicates 800 bits per inch.

3
for 9 track tape, indicates 1600 bits per inch.

Default: If the DEN subparameter is not specified:

800 bits per inch is assumed for 7 track tape;

800 bits per inch is assumed for 9 track tape without dual density;

1600 bits per inch is assumed for 9 track tape with dual density or phase-encoded drives.

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

DD

EXCP

The 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 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 Generalized Trace Facility are MODE=EXT and TRACE=USR.

The 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 values with the DSORG subparameter:

DA

indicates direct access.

IS

indicates indexed sequential.

PO

indicates partitioned organization.

PS

indicates physical sequential.

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

The MODE Subparameter

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

MODE= $\left\{ \begin{array}{c} C \\ E \end{array} \right\}$

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.

The OPTCD Subparameter

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

OPTCD=Z

All optional services must be requested by one method.

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.

The PRTSP Subparameter

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

$$\text{PRTSP} = \begin{pmatrix} 0 \\ 1 \\ 2 \\ 3 \end{pmatrix}$$

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.

DD

EXCP

The REPOS Subparameter

The REPOS subparameter specifies repositioning for tape devices.

$$\text{REPOS} = \begin{pmatrix} Y \\ N \end{pmatrix}$$

Y

indicates that repositioning is requested. A bit will be set to indicate that the user is keeping an accurate block count, and if a permanent error occurs, Dynamic Device Reconfiguration (DDR) can use the block count to reposition.

N

indicates that no repositioning is requested. DDR will not attempt repositioning.

The STACK Subparameter

The STACK subparameter specifies which stacker bin is to receive a card.

$$\text{STACK} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

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

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

The TRTCH Subparameter

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

$$\text{TRTCH} = \begin{pmatrix} C \\ E \\ T \\ ET \end{pmatrix}$$

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 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 is assumed.

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

The DCB Subparameters for GAM

You can use the following DCB subparameters with GAM:

DIAGNS
DSORG
GNCP

These subparameters are defined and the rules for coding them are explained below.

The 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 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 Generalized Trace Facility are MODE=EXT and TRACE=USR.

DD

GAM

The 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

The 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 1 is assumed.

The value of GNCP must be from 1 to 99 at execution time.

The subparameters GNCP, BFTEK, BFALIN, and HIARCHY 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 for IBM 2250 Display Unit, GC27-6971*.

The DCB Subparameter for QISAM

You can use the following DCB subparameters with QISAM:

BFALN	DIAGNS	OPTCD
BLKSIZE	DSORG	RECFM
BUFL	KEYLEN	RKP
BUFNO	LRECL	
CYLOFL	NTM	

These subparameters are defined and the rules for coding them are explained below.

The BFALN Subparameter

The BFALN subparameter specifies the boundary of each vuffer.

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

F

indicates that each buffer starts on a full-word boundary that is not also a doubleword boundary.

D

indicates that each buffer starts on a doubleword.

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

The 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. For blocks of ASCII records on magnetic tape, this maximum length is 2048 and the minimum length is 18.

BLKSIZE=number of bytes

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

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

The block size that is specified must be at least 10 bytes less than the number of data bytes available on one track of the allocated direct access device. Block size information is required only when creating a data set containing block records.

The 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



The BUFNO Subparameter

The BUFNO subparameter specifies the number of buffers to be assigned to the data control block; the maximum that can be specified in the BUFNO subparameter 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 as follows:

Method of obtaining the buffer pool:

BUILD macro instruction
GETPOOL macro instruction

Automatically

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.

The 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.

CYLOFL=number of tracks

The 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 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 Generalized Trace Facility are MODE=EXT and TRACE=USR.

The 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 data set organizations with the DSORG subparameter:

IS

indexed sequential

ISU

indexed sequential unmovable

ISU can be specified only when the data set is being created. The DSORG subparameter must be coded on the DD statement that defines the data set.

The KEYLEN Subparameter

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

KEYLEN=number of bytes

For an existing data set, the key length can only be supplied from the data set label.

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

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.

The NTM Subparameter

The NTM subparameter specifies the number of tracks to be used for a cylinder index. When the specified number of tracks has been filled, a master index is created. This information is required only when the master index option (OPTCD=M) has been selected.

NTM=number of tracks

If you omit NTM information, and OPTCD=M is specified, the master index option is ignored.

The OPTCD Subparameter

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

OPTCD= { [I] [L] [M] [R] [W] [Y] }

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.

The following values can be used with the OPTCD subparameter:

- I requests that the control program use the independent overflow areas for overflow records.
- L requests that the control program delete records that have a first byte that contains all ones. These records can be deleted when space is required for new records. This option cannot be used when RKP=0.



M

requests that master indexes be created as required, according to information specified in the NTM subparameter. This option will be ignored if the NTM subparameter is not specified.

R

requests the control program to place reorganization criteria information in the RORG1, RORG2, and RORG3 fields of the data control block.

This option is provided whenever the OPTCD subparameter is omitted from all sources.

W

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

Y

requests that the control program uses the cylinder overflow areas for overflow records.

The 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.

$$\text{RECFM} = \left\{ \begin{array}{l} \text{V} \\ \text{F} \end{array} \right. \left\{ \begin{array}{l} [\text{B}] \\ [\text{B}] \end{array} \right\}$$

The following 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, an undefined-length record is assumed with no optional features provided.

The 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.

$$\text{RKP} = \text{number}$$

Default: If you omit RKP information, a relative key position of zero is assumed.

If RPK=0 is specified for blocked fixed-length records, the key begins in the first byte of each record, and the delete option (OPTCD=L) must not be specified.

If RKP=0 is specified for unblocked fixed-length records, the key is not written in the data field; the delete option can be specified.

For variable-length records, the relative key position must be 4 or greater, when the delete option (OPTCD=L) is not specified.

The relative key position must be 5 or greater if the delete option is specified.

The DCB Subparameters for QSAM

You can use the following DCB subparameters with QSAM:

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

These subparameters are defined and the rules for coding them are explained below.

The 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 full-word boundary that is not also a doubleword boundary.

D

indicates that each buffer starts on a doubleword.

Default: If you omit BFALN, 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.

The BFTEK Subparameter

The BFTEK subparameter specifies the type of buffering to be 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.

Exchange buffering cannot be specified for variable-length blocked records or spanned records.

A

indicates record area buffering.

Default: If you omit BFTEK information, simple buffering (S) is assumed.

In the locate mode with variable-length spanned records, the control program reads and writes entire logical records rather than segments of records.

Track overflow cannot be specified in the RECFM subparameter.

DD

QSAM

The BLKSIZE Subparameter

The BLKSIZE subparameter specifies the maximum length, in bytes, of a block. The largest number that can be specified in the BLKSIZE subparameter is 32,760 except for blocks of ASCII records on magnetic tape, the largest number varies according to record format (specified by the RECFM subparameter), is 2048 and the smallest number is 18.

BLKSIZE=number of bytes

- If RECFM=F, then BLKSIZE must be logical record length.
- If RECFM=FB, then BLKSIZE must be an integral multiple of the logical record length.
- If RECFM=V, then BLKSIZE must be (maximum block size + 4).
- If RECFM=VB, then BLKSIZE must be (n X logical record length) + 4, where n is the number of logical records in the block.
- If RECFM=D or RECFM=DB, then 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 block size specified in the label.

The 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

If the BUFL subparameter is optional; if you omit it and the control program acquires buffers automatically, the block size and key length information is used to establish buffer length.

If you specify card image (MODE=C), you must specify BUFL=160.

The BUFNO Subparameter

The BUFNO subparameter specifies the number of buffers to be assigned to the data control block; the maximum normally 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 as follows:

Method of obtaining the buffer pool:

BUILD macro instruction
GETPOOL macro instruction

Automatically

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.

The 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 of the block prefix. For input, n may be any unsigned decimal number from 0 through 29. For output, n can only be 0.

L

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

DD

QSAM

The CODE Subparameter

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

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

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

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 you omit the CODE subparameter, I is assumed.

The DEN Subparameter

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

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

0

for 7 track tape, indicates 200 bits per inch.

1

for 7 track tape, indicates 556 bits per inch.

2

for 7 track tape, indicates 800 bits per inch; for 9 track tape, indicates 800 bits per inch.

3

for 9 track tape, indicates 1600 bits per inch.

Default: If the DEN subparameter is not specified:

- 800 bits per inch is assumed for 7 track tape and for 9 track tape without dual density.
- 1600 bits per inch is assumed for 9 track tape with dual density or phase-encoded drives.

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

The 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

The DSORG Subparameter

If the subparameter is not specified on the DD card, the option is not implemented. The 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 Generalized Trace Facility are MODE=EXT and TRACE=USR.

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

PSU

physical sequential, unmovable

The EROPT Subparameter

The EROPT subparameter specifies the option to be executed if an error occurs in reading or writing a record.

$$\text{EROPT} = \left\{ \begin{array}{l} \text{ACC} \\ \text{SKP} \\ \text{ABE} \end{array} \right\}$$

ACC

indicates that the system is to accept the block causing the error.

SKP

indicates that the system is to skip the block causing the error.

ABE

indicates that the system is to cause abnormal end of task.

Default: If you omit EROPT subparameter, ABE is assumed.

The FUNC Subparameter

The FUNC subparameter specifies the type of data set to be opened for the 3525 Card-Read-Punch-Print.

$$\text{FUNC} = \text{function}$$

You can code the following functions with the FUNC subparameter:

I

interpret-punch data set

R

read

P

punch

W

print

D

data protection for a punch data set

X

printer

T

two-line printer

The only valid combinations of these values are:

I	WT	RWX	PWXT
R	WXT	RWT	RPW
P	RP	RWXT	RPWY
W	RPD	PW	RPWXT
WX	RW	PWX	RPWD

Default: If the function information is not supplied by any source, P is assumed.



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

For variable-length spanned records (VS or VBS) processed under QSAM in locate mode, if the logical record size exceeds 32,756, LRECL=X must be specified.

For ASCII records on magnetic tape, the maximum record length is 2048 bytes and the minimum record length is 18 bytes.

The MODE Subparameter

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

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

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, and TRTCH are mutually exclusive. Therefore, do not code any of these other subparameters with MODE.

The OPTCD Subparameter

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

OPTCD= {
B
C
H
Q
T
U
W
Z
UC
WC
ZC

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.

The following values can be used with the OPTCD subparameter:

- B**
requests that end-of-file recognition be disregarded for tapes.
- C**
requests that chained scheduling be used. Chained scheduling will be ignored unless ADDRSPC=REAL has been specified.
- H**
requests hopper empty exit for Optical Readers.
- 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**
only for 1403 printers with the Universal Character Set feature; unblocks data checks and allows analysis by an appropriate error analysis routine.
- If U is omitted, data checks are blocked, i.e., not recognized as errors.
- 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.

DD

QSAM

The PRTSP Subparameter

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

$$\text{PRTSP} = \begin{pmatrix} 0 \\ 1 \\ 2 \\ 3 \end{pmatrix}$$

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.

The 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.

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

With 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.

The following 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.

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, i.e., no truncated blocks or unfilled tracks within the data set, with the exception of the last block or track.

for variable-length records, a record may span more than one block. Exchange buffering 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. Variable length records cannot be in ASCII.

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

The STACK Subparameter

The STACK subparameter specifies which stacker bin is to receive a card.

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

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

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

The TRTCH Subparameter

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

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

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

DD

QSAM

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 is 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.

The DCB Subparameters for TCAM

The following DCB subparameters can be used with TCAM:

BLKSIZE	BUFOUT	PCI
BUFIN	BUFSIZE	RECFM
BUFL	LRECL	RESERVE
BUFMAX	OPTCD	THRESH

These subparameters are defined and the rules for coding them are explained below.

The 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.

The 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, 1 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 can not 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.

The BUFL Subparameter

The BUFL subparameter specifies the length in bytes of each of the 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 bytes.



The BUFMAX Subparameter

The BUFMAX subparameter specifies the maximum number of buffers to be allocated to a line at one time. The number specified must be greater than 1 but may not exceed 15. 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, 2 is assumed as the value.

The BUFOUT Subparameter

The BUFOUT subparameter specifies the number of buffers to be assigned initially for sending operations for each line in the line group.

BUFOUT=number of buffers

Default: If you omit the BUFOUT subparameter, 2 is assumed.

The number specified must be less than the number of buffers in the buffer pool for this line group and may not exceed 15.

The number of buffers specified in the combined BUFIN and BUFOUT operands must be no greater than the number of buffers in the buffer pool for this line group (not including those for disk activity only.)

The BUFSIZE Subparameter

The BUFSIZE subparameter specifies the length in bytes of each of the buffers to be used for all lines in a particular line group. This length must be at least 31 bytes, but may not exceed 65,535.

The buffer size should be an even multiple of the buffer-unit size as specified in the INTRO macro instruction; the maximum number of buffer-units per buffer is 255.

BUFSIZE=number of bytes

The 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, then LRECL must be equal to the largest logical record length + 4.
- If RECFM=F or FB, then LRECL must be equal to the largest logical record length.
- If RECFM=U, then 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.

The OPTCD Subparameter

The OPTCD subparameter specifies the optional services to be 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 when used in combination, no commas are permitted between characters.

The following 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 be handle 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 is to be placed in an eight-byte field in the work area.

DD

TCAM

The PCI Subparameter

The PCI subparameter specifies whether or not a program-controlled interruption (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.

$$\text{PCI} = \left(\left[\begin{array}{c} \text{N} \\ \text{R} \\ \text{A} \end{array} \right] \quad \left[\begin{array}{c} \text{,N} \\ \text{,R} \\ \text{,A} \end{array} \right] \right)$$

The operands shown in the format above 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. A buffer is allocated in place of the freed buffer.

Default: If you omit the PCI subparameter, PCI=(A,A) is assumed.

The 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.

$$\text{RECFM} = \left. \begin{array}{c} \text{U} \\ \text{V} \\ \text{F} \end{array} \right\} [\text{B}]$$

The following 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.

The 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 to be reserved in the first buffer that receives an incoming message.
- number 2**
indicates the number of bytes to be reserved in all the buffers following the first buffer in a multiple-buffer header situation.

The THRESH Subparameter

The THRESH subparameter specifies the percentage of the non-reusable disk message queue records to be used before a flush closedown occurs.

$$\text{THRESH} = \text{number}$$

If you omit the THRESH subparameter, closedown occurs when 95% of the records have been used.

The 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.

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. Do not use the DDNAME parameter to refer to a DD statement that has DYNAM coded on it.
6. A DD statement to which a DDNAME parameter refers cannot contain any reference to a DD statement that follows the one with the DDNAME parameter.

Examples of the DDNAME Parameter

```
//STEP1 EXEC PGM=PROGRAM8
//DD1 DD DDNAME=INPUT
//DD2 DD DSNAME=WELL,DISP=OLD
```

The above 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
```

DD

```
//DD5      DD      DSNAME=ADDN,DISP=( ,PASS ),UNIT=2400
//ST5      EXEC     PGM=FINE
//DD6      DD      DSNAME=*.ST4.DD1,DISP=(OLD,KEEP)
```

The DD statement named DD5 defines the data set for the statement named DD1. The DD statement of the second job step 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      DSNAME=A.B.C,DISP=OLD
//          DD      DSNAME=LEV.FIVE,DISP=OLD
//          DD      DSNAME=LEV.FIVE,DISP=OLD
//SKIP     DD      DSNAME=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, will be 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.

The DEST Parameter (For OS/VS1 with RES)

Keyword, Optional

In VS1, remote entry services (RES) provide 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/VS1 RES Workstation User's Guide*, GC28-6879.

```
DEST=userid
```

userid

indicates a remote destination for an output data set.

DD

Rules for Coding

1. Code a valid userid; valid userids are established by your installation. They must consist of one to seven alphameric 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 user-id STAT04 receives the output described by the GWB DD statement.

The 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 terminates abnormally (conditional disposition). For further information on the DISP parameter, see *OS/VS JCL Services*, GC28-0617.

$$\text{DISP}=(\begin{array}{l} \text{NEW} \\ \text{OLD} \\ \text{SHR} \\ \text{MOD} \end{array} \left[\begin{array}{l} \text{,DELETE} \\ \text{,KEEP} \\ \text{,PASS} \\ \text{,CATLG} \\ \text{,UNCATLG} \end{array} \right] \left[\begin{array}{l} \text{,DELETE} \\ \text{,KEEP} \\ \text{,CATLG} \\ \text{,UNCATLG} \end{array} \right])$$

NEW

specifies that the data set is to be created in this job step.

OLD

specifies that the data set existed before this job step.

SHR

specifies that the data set existed before this job step and can be used simultaneously (shared) by another job, since it will only be read.

MOD

specifies that the read/write mechanism is to be positioned after the last record in the data set, and, if the system cannot find volume information for the data set, specifies that the data set is to be created.

,DELETE

specifies that the data set is no longer needed and its space on the volume is to be released at the end of this job step for use by other data sets.

,KEEP

specifies that the data set is to be kept on the volume at the end of this job step.

,PASS

specifies that the data set is to be passed for use by a subsequent job step in the same job.

,CATLG

specifies that the data set is to be kept at the end of this job step and an entry pointing to the data set is to be placed in the system catalog.

,UNCATLG

specifies that the data set is to be kept at the end of this job step but the entry pointing to the data set in the system catalog is to be removed.

specifies 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 to be released for use by other data sets if this step abnormally terminates.



,KEEP

specifies that the data set is to be kept on the volume if this step abnormally terminates.

,CATLG

specifies that an entry pointing to the data set is to be placed in the system catalog if this step abnormally terminates.

,UNCATLG

specifies that the entry pointing to the data set in the system catalog is to be removed if this step abnormally terminates.

Rules for Coding

1. If you code only the first subparameter, you need not enclose it in parentheses.
2. If the data set is new, you can omit the subparameter NEW. However, if you specify a disposition or conditional disposition, you must code a comma to indicate the absence of NEW.
3. You can omit the DISP parameter if a data set is created and deleted during a job step.
4. If you do not want to change the automatic disposition processing performed by the system, you need not code the second subparameter. (When the second subparameter is not coded, the system automatically keeps data sets that did exist before the job and automatically deletes data sets that did not exist before the job.) If you omit the second subparameter and code a conditional disposition, you must code a comma to indicate the absence of the second subparameter.
5. The DISP, SYSOUT, and DDNAME parameters are mutually exclusive parameters; therefore, when SYSOUT or DDNAME is coded, do not code the DISP parameter.
6. You must specify a disposition of PASS or DELETE for a data set with a system-generated name, i.e. when, DSNNAME=dsnname is omitted from the DD statement. Any other disposition will be overridden by the system with PASS.

Examples of the DISP Parameter

```
//DD DD DSNNAME=D99.GROUP.SIX,UNIT=2314,VOLUME=SER=111111, X  
// DISP=(NEW,CATLG,DELETE),SPACE=(TRK,(5,1))
```

This DD statement defines a new data set and requests the system to create an index entry in the system catalog that points to this data set if the step terminates normally. Because the data set's name is qualified, you must use the IEHPROGM utility program to create the indexes in the catalog for D99 and GROUP before you request the system to catalog the data set.

```
//DD2 DD DSNNAME=FIX,UNIT=2400-1,VOLUME=SER=44889, X  
// DISP=(OLD,,DELETE)
```

This DD statement defines an existing data set and implies that the data set is to be kept if the step terminates normally. (For an existing data set, the system assumes it is to keep the data set if no disposition is specified.) The statement requests that the system delete the data set if the step abnormally terminates.


```

//STEP1      EXEC      PGM=FULL
//DD1        DD        DSNAME=SWITCH.LEVEL18.GROUP12,UNIT=2314,
//           DD        VOLUME=SER=LOCAT3,SPACE=(TRK,(80,15)),DISP=(,PASS) X
//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 STEP 1 defines a new data set and requests that the data set be passed. If STEP1 abnormally terminates, the data set will be deleted since it is a new data set and a conditional disposition was not specified. The DD statement named DD3 in STEP2 receives the passed data set and requests that the data set be passed. If STEP2 abnormally terminates, the data set will be deleted because of the conditional disposition of DELETE. The DD statement named DD4 in STEP3 receives the passed data set and requests that the data set be cataloged at the end of the step. If STEP3 abnormally terminates, the data set will be deleted because of the conditional disposition of DELETE.

DD

The 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 on the delimiter statement in this publication.

```
DLM=delimiter
```

delimiter

two characters that will 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, you must code each ampersand or apostrophe as two consecutive ampersands or apostrophes.
4. The DLM parameter has meaning only on statements defining data in the input stream (DD* and DD DATA statements).
5. If you do code the DLM parameter on a DD* or DD DATA statement, the characters you assign as delimiter override 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 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 parameter for the data defined in the input stream by DD1.

The DSNNAME Parameter

Keyword, Optional

The DSNNAME 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.

$$\left. \begin{array}{l} \{ \text{DSNAME} \} \\ \{ \text{DSN} \} \end{array} \right\} = \left. \begin{array}{l} \text{dsname} \\ \text{dsname}(\text{member name}) \\ \text{dsname}(\text{generation number}) \\ \text{dsname}(\text{area name}) \\ \text{\&\&dsname}(\text{member name}) \\ \text{\&\&dsname}(\text{area name}) \\ *.ddname \\ *.stepname.ddname \\ *.stepname.procstepname.ddname \end{array} \right\}$$

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.)

\&\&dsname

specifies the name you want assigned to a temporary data set.

\&\&dsname(member name)

specifies the name you want assigned to a temporary partitioned data set and to a member within that data set.

\&\&dsname(area name)

specifies the name you want assigned to a temporary indexed sequential data set and identifies an area of that data set (INDEX, PRIME, or OVFLOW.)

***.ddname**

specifies that the data set name is to be copied from the named DD statement, which is an earlier DD statement in the job step.

***.stepname.ddname**

specifies that the data set name is to be copied from an earlier DD statement named ddname, which appears in an earlier step named stepname in the same job.

***.stepname.procstepname.ddname**

specifies that the data set name is to be copied from an earlier DD statement in a cataloged procedure. Stepname is the name of the job step that calls the procedure, procstepname is the name of the procedure step that includes the named DD statement, and ddname is the name of the DD statement that contains the data set name.

Rules for Coding

1. An unqualified data set name may consist of 1 to 8 characters. The first character must be an alphabetic or national (@,\$,#) character; the remaining characters can be any alphameric or national characters, a hyphen, or a plus zero (12-0 punch). A temporary data set name can consist of 1 through 8 characters, 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 need not code the DSNNAME parameter if the data set is created and deleted in the job, i.e., if the data set is temporary.
4. The DSNNAME and DDNAME parameters are mutually exclusive parameters; therefore, when the DDNAME parameter is coded, do not code the DSNNAME parameter.

Examples of the DSNNAME Parameter

```
//DD1 DD DSNNAME=ALPHA,DISP=(,KEEP), X  
// UNIT=2400,VOLUME=SER=389984
```

This DD statement defines a new data set whose name is ALPHA. Later job steps or jobs may retrieve this data set by supplying the data set name in the DSNNAME parameter, unit information in the UNIT parameter, and volume information in the VOLUME parameter.

```
//DD2 DD DSNNAME=PDS( PROG12 ),DISP=(OLD,KEEP),UNIT=2314, X  
// VOLUME=SER=882234
```

This DD statement retrieves a member of a partitioned data set named PDS.

```
//DD3 DD DSNNAME=εεWORK,UNIT=2400
```

This DD statement defines a temporary data set. Since the data set is to be deleted at the end of the job step, the DSNNAME parameter can be omitted. However, it may be included to facilitate a later reference to a passed data set, e.g. DSNNAME=εεWORK,DISP=OLD.

```
//STEP1 EXEC PGM=CREATE  
//DD4 DD DSNNAME=εεISDATA(PRIME),DISP=(,PASS),UNIT=(2311,2), X  
// 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. Since the temporary data set will be passed when it is defined in STEP1, STEP2 can retrieve the data set.

The FCB Parameter

Keyword,Optional

The FCB parameter specifies the forms control image to be used to print an output data set on a 3211 printer, or a 3525 card punch with the read feature.

For further information on the forms control buffer, see *OS/VS Data Management for System Programmers*, GC28-0631.

```
FCB=( image-id [ ,ALIGN  
               [ ,VERIFY ] ] )
```

image-id

the code that identifies the image to be loaded into the forms control buffer.

,ALIGN

requests the operator to check the alignment of the printer forms before the data set is printed.

,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 1 to 4 alphameric and national (#, @, \$) characters. The first character must be an alphabetic or national character.
2. If you omit the FCB parameter, the default image is used if it is currently in the buffer. Otherwise, the operator will be requested to specify an image.
3. The FCB parameter will be ignored if the data set is not written to a 3211 printer, or to a 3525 card punch with the read feature.
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.

Examples of the FCB Parameter

```
//DD1 DD UNIT=3211,FCB=(IMG1,VERIFY)
```

This DD statement defines the output data set that is to be written to a 3211 printer. The FCB parameter requests that the data set be written using the control information corresponding to the forms control image with the code IMG1. Since VERIFY is coded, the forms control image will be displayed on the printer before the data set is printed and the operator will be asked to align the printer forms.

```
//DD2 DD SYSOUT=A,FCB=IMG2
```

This DD statement defines an output data set that is to be written to the device that corresponds with class A. The FCB parameter will be ignored if the device is not a 3211 printer.

The HOLD Parameter (For OS/VS1 with RES)

Keyword, Optional

In VS1, remote entry services (RES) provide the facility to submit jobs to a central computing center from a remote workstation and to route output to remote workstations.

The HOLD parameter specifies that an output data set is to be held on a queue until the central operator or the user to which the data is being sent issues a ROUTE command.

For further information on RES, see *OS/VS1 Workstation User's Guide*, GC28-6879.

$$\text{HOLD} = \left\{ \begin{array}{l} \text{YES} \\ \text{NO} \end{array} \right\}$$

YES

specifies that processing of the output data set is to be deferred until a ROUTE 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 on the normal output queue.
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 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 will be held on a queue until the user identified by STAT04 or the central operator issues a ROUTE command.

The 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 ] [ ,NOPWREAD ] [ ,IN ] [ ,OUT ] [ , ] [ EXPDT=yyddd ] [ 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 has 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 can supply the system with the correct password, i.e., the data set cannot be read, changed, extended, or deleted.

,NOPWREAD

specifies that the data set can be read without the password, but the operator must give the password before the data set can be changed, extended, or deleted.

specifies that another subparameter follows and, for a new data set, the data set is not to be password protected.

,IN

specifies that the data set is to be processed for input only.

,OUT

specifies that the data set is to be processed for output only.

specifies that either the RETPD or EXPDT subparameter follows and one or more subparameters precede it.

EXPDT=yyddd

specifies the date when the data set can be deleted or overwritten by another data set. Assign a 2-digit year number and a 3-digit day number.

RETPD=nnnn

specifies the length of time in days that the data set must be kept. Assign the number of days that must pass before the data set can be deleted or overwritten by another data set.

Rules for Coding

1. All the subparameters except the last subparameter in the LABEL parameter are positional subparameters. Therefore, if you 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. If the data set has IBM standard labels, you can omit the subparameter SL.
4. When you are defining a data set that resides or will reside on a direct access volume, only SUL or SL can be specified as the second subparameter.
5. If you are processing ASCII data on unlabeled (NL) tapes, you must code OPTCD=Q in your DCB macro instruction or on your DD statement.
6. The LABEL, DDNAME, and SYSOUT parameters are mutually exclusive parameters; therefore, if DDNAME or SYSOUT is coded, do not code the LABEL parameter.
7. 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**).

Examples of the LABEL Parameter

```
//DD1 DD DSN=HERBI,DISP=(NEW,KEEP),UNIT=TAPE, X
//      VOLUME=SER=T2,LABEL=(3,NSL,RETPD=188)
```

This DD statement defines a new data set. The LABEL parameter tells the system: (1) this data set is to be the third data set on the tape volume; (2) this tape volume has nonstandard labels; (3) this data set is to be kept for 188 days.

```
//DD2 DD DSN=A.B.C,DISP=(,CATLG,DELETE),UNIT=2400-2, X
//      LABEL=(,NL)
```

This DD statement defines a new data set and requests the system to catalog it. The catalog entry for this data set will not indicate that the data set has no labels. Therefore, each time this data set is referred to by a DD statement, the statement must include LABEL=(,NL).

```
//DD3 DD DSN=SPECS,UNIT=2400,VOLUME=SER=10222, X
//      DISP=OLD,LABEL=4
```

This DD statement defines an existing data set. The LABEL parameter indicates that the data set is the fourth data set on the tape volume.

```
//STEP1 EXEC PGM=FIV
//DDX DD DSN=CLEAR,DISP=(OLD,PASS),UNIT=2400-4, X
//      VOLUME=SER=1257,LABEL=(,NSL)
//STEP2 EXEC PGM=BOS
//DDY DD DSN=*.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 since this information is available to the system; the label type is not available to the system and must be coded.

The OUTLIM Parameter (For OS/VS1)

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 provided by the System Management Facilities option is taken to a user supplied routine that determines whether to cancel the job or increase the limit. If the exit routine is not supplied, the job is cancelled.

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 16777215.

DD

Rules for Coding

1. The OUTLIM parameter has meaning only if the System Management Facilities option is in use in the system and job, and step data collection was selected at system generation.
2. **Default:** If you do not specify the OUTLIM parameter, the system default will be used unless SYSUDUMP or SYSABEND is being processed. If SYSUDUMP or SYSABEND is being processed, or if OUTLIM = 0, no output limiting will be done.
3. The OUTLIM parameter is ignored unless SYSOUT is coded in the operand field of the same DD statement.
4. The value specified for OUTLIM can be any number from 1 through 16777215.
5. If you omit OUTLIM or if you code OUTLIM=0, no output limiting is done.
6. **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.)
7. The OUTLIM and DDNAME parameters are mutually exclusive. Do not code them together on one DD statement.
8. In OS/VS2 the OUTLIM parameter is ignored.

Example of the OUTLIM Parameter

```
//OUTPUT DD SYSOUT=F,OUTLIM=1000
```

The limit for the number of logical records is 1000.

The 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 1 through 8 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 will supply information that was not supplied in the DCB macro instruction for the data control block.

The 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 need not enclose it in 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. In OS/VS1 with the I/O Load Balancing option in use, and in OS/VS2 the SEP parameter will be ignored.

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,           X
//          SPACE=(CYL,(3,1))
//DD3  DD         DSNAME=NABS,DISP=OLD,VOLUME=SER=7110,UNIT=2314
//DD4  DD         DSNAME=PARE,DISP=OLD,VOLUME=SER=E59,           X
//          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. Since the SEP parameter did not include the ddname DD1, the data set defined by DD1 and the data set defined by DD4 may or may not be assigned to the same channel.

The SPACE Parameter

Keyword,Optional

$$\left. \begin{array}{l} \text{SPACE}=(\left\{ \begin{array}{l} \text{TRK} \\ \text{CYL} \\ \text{blocklength} \end{array} \right\}, (\text{primary quantity} \left[\begin{array}{l} , \\ \text{secondary quantity} \end{array} \right] \left[\begin{array}{l} , \\ \text{directory} \end{array} \right] \left[\begin{array}{l} , \\ \text{RLSE} \end{array} \right] \left[\begin{array}{l} , \\ \text{CONTIG} \\ , \\ \text{MXIG} \\ , \\ \text{ALX} \end{array} \right] \left[\begin{array}{l} , \\ \text{ROUND} \end{array} \right] \right\}) \\ \text{SPACE}=(\text{ABSTR}, (\text{primary quantity}, \text{address} \left[\begin{array}{l} , \\ \text{directory} \end{array} \right] \left[\begin{array}{l} , \\ \text{index} \end{array} \right] \right\}) \end{array} \right\}$$

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 Appendix C of this publication.

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 allocation can be done up to 16 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 that space allocated to the data set that is not used when the data set is closed, is to be released.

, 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 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, e.g., 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 blocklength 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:** There must be enough available space on one volume to satisfy the primary quantity. If you request that a particular volume be used and there is not enough space available on 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 is terminated.

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 10 of this publication.

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 do specify a secondary quantity and the data set requires additional space, the system allocates this space based on the quantity you specified. The system attempts to allocate the secondary quantity in contiguous tracks or cylinders. If contiguous space is not available, the system attempts to allocate the secondary quantity in up to five noncontiguous blocks (extents) of space.

Each time the data set requires more space, the system allocates the secondary quantity. This space is allocated on the same volume on which the primary quantity was allocated until: (1) there is not enough space available on the volume to allocate the secondary quantity, or (2) a total of 16 extents have been allocated to the data set. If either of these conditions is satisfied, the system must allocate the secondary quantity on another volume.

You can specify 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 non-specific volume request, code PRIVATE and specify more than one volume in the VOLUME parameter.

If there is no more space available on those volumes that you requested, if at least one volume is demountable, the system will request scratch volumes to be mounted until either the data set is complete or until all entries in the JFCB are filled. If the entries in the JFCB are already filled or if there is no demountable volume, the job step will abend.

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.

Examples of the SPACE Parameter

```
//DD1 DD DSNAME=TEMP,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 will be ignored; if a direct access volume is assigned, the SPACE parameter will be used to allocate space to the data set. The SPACE parameter includes only the required subparameters (i.e., the type of units and a primary quantity), and requests the system to allocate 10 cylinders.

```
//DD2 DD DSNAME=ELLN,DISP=(,KEEP),UNIT=2314, X
// VOLUME=SER=11257,SPACE=(1024,(100,25),,,ROUND), X
// DCB=BLKSIZE=2048
```

This DD statement defines a new data set that will be written on a direct access volume. The SPACE parameter requests that the system compute the space required for the primary quantity; the system will compute the space required based on an average block length of 1024 bytes, and up to 100 blocks of data will be written. If more space is required, the system will compute how much additional space is to be allocated; the system will compute the space required based on a maximum block length of 2048 bytes (specified in the BLKSIZE subparameter), and up to 25 blocks of data will be written. Since the ROUND subparameter is coded, the system will ensure 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, X
// VOLUME=SER=26143,SPACE=(TRK,(200,,10),,CONTIG)
```

This DD statement defines a new partitioned data set. The system will allocate 200 tracks to the data set and ten 256-byte records for a directory. Since the CONTIG subparameter is coded, the system will allocate 200 contiguous tracks on the volume.

```
//DD4 DD DSNAME=INDSEQ(INDEX),UNIT=2314,DCB=DSORG=IS, X
// DISP=(,KEEP),SPACE=(ABSTR,(20,40))
```

This DD statement defines the index are for a new indexed sequential data set. The SPACE parameter will allocate 20 tracks (for a 2314, 20 tracks equal 1 cylinder), beginning with the fortieth track on the volume (the fortieth track on the volume is the beginning of the third cylinder).

The SPLIT Parameter

Keyword,Optional

The SPLIT parameter allocates space to two or more new data sets that are to share cylinders. For further information on the SPLIT parameter, see *OS/VS JCL Services*, GC28-0617.

$$\text{SPLIT}=\left\{ \begin{array}{l} (n,\text{CYL},(\text{primary quantity}[, \text{secondary quantity}])) \\ n \\ (\text{percent}, \text{block length}, (\text{primary quantity}[, \text{secondary quantity}])) \\ \text{percent} \end{array} \right\}$$

n
the number of tracks per cylinder you want allocated to the first data set.

CYL
specifies that space is to be allocated by cylinder.

primary quantity
specifies how many cylinders are to be allocated for use by all the associated data sets.

,secondary quantity
specifies how many more cylinders are to be allocated to a data set if additional space is required.

n
the number of tracks per cylinder you want allocated to the data set defined on the DD statement.

percent
the percentage of tracks per cylinder you want allocated to the first data set, a number from 1 through 99.

block length
specifies the average block length of the data. The system computes how many cylinders to allocate.

primary quantity
specifies the total number of blocks to be allocated for use by all the associated data sets.

,secondary quantity
specifies how many more blocks are to be allocated to a data set if additional space is required.

percent
the percentage of tracks per cylinder you want allocated to the data set defined on the DD statement.

Rules for Coding

1. 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. You need not code volume and unit information on the following DD statements that contain the SPLIT parameter.
6. If you do not specify a secondary quantity, you need not enclose the primary quantity in parentheses.
7. **Blocklength:** The average blocklength cannot exceed 65,535 bytes. If the blocks have keys, code the DCB subparameter KEYLEN on the DD statement and specify the keylength.

Examples of the SPLIT Parameter

```
//STEP1 EXEC      PGM=CREATE
//DD1  DD         DSNAME=QUEST,DISP=( ,KEEP ),UNIT=2314,          X
//         VOLUME=SER=757500,SPLIT=( 7,CYL,( 30,1 ))
//DD2  DD         DSNAME=APP,DISP=( ,KEEP ),SPLIT=8
//DD3  DD         DSNAME=SET,DISP=( ,KEEP ),SPLIT=5
```

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 4 tracks per cylinder to this data set. The DD statement named DD3 requests that the system allocate 3 tracks per cylinder to this data set.

```
//STEP2 EXEC      PGM=PAGE
//DDX  DD         DSNAME=ISSA,DISP=( ,KEEP ),UNIT=2314,          X
//         VOLUME=SER=49463,SPLIT=( 18,1024,( 700 ))
//DDY  DD         DSNAME=SEL12,DISP=( ,KEEP ),SPLIT=48
//DDZ  DD         DSNAME=SEVE,DISP=( ,KEEP ),SPLIT=34
```

This job step contains a sequence of DD statements that define new data sets and request that these data sets share the same cylinders. The first DD statement of the sequence, named DDX, specifies in the SPLIT parameter: (1) 18 per cent of the tracks per cylinder are to be allocated to this data set; (2) the system is to compute how many cylinders are to be allocated for use by all the data sets based on an average block length of 1024 bytes and 700 blocks are required. The DD statement named DDY requests that the system allocate 48 per cent of the tracks per cylinder to this data set. The DD statement named DDZ requests that the system allocate 34 per cent of the track per cylinder to this data set. Since the first DD statement in the sequence does not specify a secondary quantity, the job will be abnormally terminated if any of the data sets exceeds its allocated space.

The 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.

For further information on the SUBALLOC parameter, see OS/VS JCL Services, GC28-0617.

$$\text{SUBALLOC} = \left(\left\{ \begin{array}{l} \text{TRK} \\ \text{CYL} \\ \text{blocklength} \end{array} \right\} , (\text{primary quantity} [, \text{secondary quantity}] [, \text{directory}]) \left\{ \begin{array}{l} \text{,ddname} \\ \text{,stepname.ddname} \\ \text{,stepname.procstepname.ddname} \end{array} \right\} \right)$$

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 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, DDNAME, and SYSOUT parameters are mutually exclusive parameters; therefore, when SPACE, SPLIT, 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         DSNAME=DUM,DISP=( ,KEEP ),UNIT=2305-2,          X
//                                     VOLUME=SER=ALLDS,SPACE=(CYL,50, ,CONTIG)
//STEP2 EXEC      PGM=BSPED
//DD2  DD         DSNAME=SPEC50,DISP=( ,KEEP ),                  X
//                                     SUBALLOC=(CYL,(20,1),STEP1.DD1)
//DD3  DD         DSNAME=SPEC51,DISP=( ,KEEP ),                  X
//                                     SUBALLOC=(TRK,(44,7),STEP1.DD1)
//DD4  DD         DSNAME=SPEC52,DISP=( ,KEEP ),                  X
//                                     SUBALLOC=(CYL,25,STEP1.DD1)
```

The data set from which space is to be suballocated is defined on the DD statement named DD1 in STEP1. Fifty cylinders will be allocated to the data set and the cylinders will be contiguous. The DD statements named DD2, DD3, and DD4 in STEP2 request a portion of this space in the SUBALLOC parameter by referring the system to the data set defined on the DD statement named DD1 in STEP1. The order of the data sets on the volume, because of the request for suballocation, will be DUM, SPEC50, SPEC51, and SPEC52.

```
//STEPX EXEC      PGM=GARV
//DD5  DD         DSNAME=SIMP,DISP=( ,KEEP ),UNIT=2314,          X
//                                     VOLUME=SER=315046,SPACE=(CYL,100, ,CONTIG)
//DD6  DD         DSNAME=FIELD,DISP=( ,KEEP ),
//                                     SUBALLOC=(1024,(800,60),DD5)
//STEPY EXEC      PGM=BERSS
//DD7  DD         DSNAME=PDS,DISP=( ,KEEP ),                      X
//                                     SUBALLOC=(CYL,(75, ,8),STEPX.DD5)          X
```

The data set from which space is to be suballocated is defined on the DD statement named DD5 in STEPX. One hundred cylinders will be allocated to the data set and the cylinders will be contiguous. The DD statement named DD6 requests a portion of this space in units of blocks. The system will compute 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.

The 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/VS JCL Services*, GC28-0617.

```
SYSOUT=(classname [ ,program name ] [ ,form number ] )
```

classname

specifies the class associated with the output device to which you want your output data set written.

,program name

specifies the member name of a program in the system library that is to write your output data set, instead of the system output writer, to a unit record device.

,

specifies that the system output writer is to write your output data set to a unit record device, and a form number follows.

,form number

specifies that the output data set should be printed or punched on a special output form.

DD

Rules for Coding

1. The classname can be any alphanumeric character (A-Z, 0-9).
2. The form number is 1 to 4 alphanumeric and national (@,\$,#) characters.
3. If you omit program name and form number, you need not enclose the classname in parentheses.
4. The UNIT, SPACE, OUTLIM, UCS, FCB, COPIES, and DCB parameters can be coded with the SYSOUT parameter. Besides the mutually exclusive parameters listed below, other parameters coded with the SYSOUT parameter are ignored.
5. The DISP, DDNAME, AFF, SEP, VOLUME, LABEL, SPLIT, and SUBALLOC parameters and the SYSOUT parameter are mutually exclusive parameters; therefore, if any of these parameters are coded, do not code the SYSOUT parameter.

Examples of the SYSOUT Parameter

```
//DD1 DD SYSOUT=P
```

This DD statement specifies that the data set is to be written to the device corresponding to class P. Since the UNIT and SPACE parameters are not coded, the system will obtain device and space allocation information from the input reader procedure.

```
//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 to be written to the device corresponding to class C. The DCB parameter is coded to complete the data control block associated with this data set. Since the classnames in the SYSOUT parameter and the

MSGCLASS parameter on the JOB statement are the same, the system messages resulting from this job and the output data set will be written to the same unit record device.

```
//DD5 DD SYSOUT=(F,,7402)
```

This DD statement specifies that the data set is to be written to the device corresponding to class F and the output data set is to be printed on a special form. The form number is 7402.

The TERM Parameter (For OS/VS1 with RES)

Keyword, Optional

In VS1, remote entry services (RES) provide 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 the presence of an RTAM (Remote Terminal Access Method used with RES) device to the system.

For further information on RES see *OS/VS1 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 for OS/VS1 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
```

The TERM Parameter (For OS/VS2 with TSO)

Keyword, Optional

The TERM parameter notifies the operating system that a data set is coming from or going to a time sharing terminal.

```
TERM=TS
```

TS

indicates to the system that the input or output data being defined is coming from or going to a time sharing terminal.

Rules for Coding

1. The TERM parameter is effective only in OS/VS2 using TSO. In batch processing or in OS/VS2, the TERM parameter will be checked for syntax; the DD statement containing the TERM parameter will be treated as a DD DUMMY statement.
2. TS is the only value that you can specify with the TERM parameter. If you use any other value, you will receive a JCL error message.
3. You can concatenate a DD statement with a DD statement that contains TERM=TS only if it is the last DD statement in a job step.
4. Code only the DCB parameter with the TERM parameter. Any other parameters coded on a DD statement with TERM are ignored.

Examples of the TERM Parameter

```
//DD1      DD      TERM=TS  
or  
//DD2      DD      UNIT=2400,DISP=(MOD,PASS),TERM=TS
```

The above two DD statements are equivalent in effect. In the time sharing environment, all the parameters coded on the second DD statement are ignored except the TERM parameter. In a batch processing environment, the UNIT and DISP parameters are used but TERM is ignored.

```
//DD3      DD      UNIT=2400,DISP=(MOD,PASS),DCB=(LRECL=80,BLKSIZE=80),  
              TERM=TS,LABEL=(,NL)
```

In a time sharing environment, all the parameters in the above example except TERM and DCB are ignored.

The UCS Parameter

Keyword, Optional

The UCS parameter describes the character set to be used for printing an output data set on a 1403 or 3211 printer.

For further information on the UCS parameter, see *OS/VS Data Management for System Programmers*, GC28-0631.

```
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*, GA24-3112. The fold mode is most often requested when uppercase and lowercase 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 will be 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 universal character set (UCS) feature, the UCS parameter will be 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, 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

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. Since VERIFY is coded, the character set image will be displayed on the printer before the data set is printed.

```
//DD2 DD SYSOUT=G,UCS=PCHN
```

This DD statement defines an output data set that is to be written to the unit record device that corresponds with class G. If the device is a printer with the universal character set, the request in the UCS parameter for the special character set with the code PCHN will be recognized. Otherwise, the UCS parameter will be ignored.

DD

The 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/VS JCL Services*, GC28-0617.

$$\left. \begin{array}{l} \text{UNIT}=(\left[\begin{array}{l} \text{unit address} \\ \text{device type} \\ \text{group name} \end{array} \right] \left[\begin{array}{l} \text{,unit count} \\ \text{,P} \\ \text{,} \end{array} \right] [\text{,DEFER}] [\text{,SEP}=(\text{ddname}, \dots)]) \\ \text{UNIT}=\text{AFF}=\text{ddname} \\ \text{UNIT}=\text{symbolic address} \end{array} \right\}$$

Note: This use of the UNIT parameter is restricted to OS/VS1 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. Specify the device number.

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. In OS/VS1 with the I/O Load Balancing option in use, and in OS/VS2, the SEP subparameter will be ignored.

(ddname,...)

the names of up to eight earlier DD statements in the job step that define data sets from which you want unit separation.

AFF=

indicates that unit affinity is desired. The system will assign the data set to the same device(s) as assigned to the data set specified by the ddname that follows.

ddname

the name of an earlier DD statement in the job step that defines a data set with which you want unit affinity.

symbolic address (OS/VS1 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
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 identify a telecommunications device by its unit address, the system will allocate that device on a shared basis whether or not the device is already allocated.

4. **Group name:** A group name is 1 through 8 alphanumeric characters and can identify a device or a group of devices. The group can consist of devices of the same type or difference direct access and tape device types.

When you code a group name, you allow the system to assign any available device from the group. If a group consists of only one device, the system will assign 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 may not necessarily be of that same group.

When the automatic volume recognition feature (AVR) is included in the system and you specify a group name, this feature will assign devices to volumes already mounted, but will not request mounting of any volume that is not mounted.

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 will assign one device.

If you receive a passed data set or refer the system to a cataloged data set or earlier DD statement for volume and unit information (VOLUME=REF=reference), the system will assign one device, even if more devices were requested in an earlier DD statement.

Only in one case will the system assign 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 will assign 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 devices to become available, or if the request for unit separation should be ignored, or if the job should be cancelled. When you make a nonspecific volume request

for a data set and request unit separation for that data set, the request may be ignored, depending on how many disk drives are available and how much space is available on those disk drives. A message will not be issued in this case if unit separation cannot be satisfied.

9. UNIT=AFF and DISP=NEW are mutually exclusive on the same DD statement. Therefore, if your data set is new, you cannot request unit affinity to another ddname.

Examples of the UNIT Parameter

```
//STEP2 EXEC PGM=POINT
//DDX DD DSNAME=EST,DISP=MOD,VOLUME=SER=(42569,42570), X
// UNIT=(2314,2)
//DDY DD DSNAME=ERAS,DISP=OLD,UNIT=2400-2
//DDZ DD DSNAME=RECK,DISP=OLD, X
// VOLUME=SER=(40653,13262),UNIT=AFF=DDX
```

The DD statement named DDZ requests that the system assign the same unit to this data set that it assigns to the data set defined on the statement named DDX. Since DDX requests two devices, these two devices are assigned to the data set defined on DDZ.

```
//DD1 DD DSNAME=AAG3,DISP=(,KEEP), X
// VOLUME=SER=13230,UNIT=2400
```

This DD statement defines a new data set and requests the system to assign any 2400 9-track tape drive to the data set.

```
//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 will be obtained from the catalog.

```
//DD3 DD DSNAME=COLLECT,DISP=OLD, X
// VOLUME=SER=1095,UNIT=(DISK,,DEFER)
```

This DD statement defines an existing data set that resides on a direct access volume and requests the system to assign any device that is part of the group named DISK. Since DEFER is coded, the volume will not be 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.

The 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/VS JCL Services*, GC28-0617.

```
{ VOLUME } = ( [PRIVATE] [ ,RETAIN ] [ ,volume sequence number ] [ ,volume count ] ,  
{ VOL } [ SER=(serial number,...)  
REF=dsname  
REF=*.ddname  
REF=*.stepname.ddname  
REF=*.stepname.procstepname.ddname
```

PRIVATE

indicates that no output data set can be allocated to this volume unless the volume is specifically requested, and the volume is to be demounted after its last use in the job step, unless RETAIN is coded or the data set is passed.

,RETAIN

indicates that this volume is not to be demounted after its last use in the job step.

indicates that the PRIVATE 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 you want used to begin processing.

indicates that you want to begin processing of an existing multivolume data set with the first volume, and the volume count subparameter follows.

,volume count

specifies the maximum number of volumes an output data set requires.

specifies that either the SER or REF subparameter follows and one or more subparameters precede it.

SER=

indicates that serial numbers of the volumes on which the data set resides or is to reside, are specified.

(serial number,...)

the serial numbers of the volumes on which the data set resides or will reside.

REF=

indicates that the serial numbers of the volumes on which the data set resides or is to reside are identified on an earlier DD statement in the job or in the catalog.

dsname

the name of a cataloged or passed data set. The system locates the information about the data set and assigns your data set to the same volumes as are assigned to the cataloged or passed data set.



- *.ddname
specifies that the system must obtain the volume serial numbers from an earlier DD statement named "ddname" in the same job step.
- *.stepname.ddname
specifies that the system must obtain the volume serial numbers from a DD statement named "ddname," which was defined in an earlier job step named "stepname."
- *.stepname.procstepname.ddname
specifies that the system must obtain the volume serial numbers from a DD statement named "ddname," which was defined in an earlier procedure step named "procstepname"; the procedure step is part of a procedure that was called by an earlier job step named "stepname."

Rules for Coding

1. The VOLUME, 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, i.e., 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. **Non-specific request:** To make a nonspecific volume request, you can code the PRIVATE subparameter, or the PRIVATE and RETAIN subparameters, and the volume count subparameter in the VOLUME parameter. You should not code the volume sequence number subparameter to make a nonspecific volume request.
4. **The PRIVATE subparameter:** If you code only PRIVATE, you need not enclose it in 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 need not code the RETAIN subparameter when the data set is to be passed; the system will automatically retain 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.

Normally, you code a volume sequence number when you have not specified volume serial numbers on the DD statement (i.e., 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 will begin processing with the volume that corresponds with the volume sequence number.
7. **The volume count:** The volume count value can range from 1 to 255.

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 will request scratch volumes to be mounted until either the data set is complete or until all entries in the JFCB are filled. If the JFCB entries are already filled, or the volumes is not demountable, the job step will abend. If you have specified a volume count greater than 99, duplicate volume serial numbers will be 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 can ensure that the data set can be written on more than one volume if it exceeds one volume.

8. **The volume serial number:** You can specify a maximum of 255 volume serial numbers per DD statement and a maximum of 4095 volume serial numbers per job step.

A volume serial number must be 1 to 6 characters in length. If the number is less than 6 characters, it will be 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 alphanumeric characters.

The SER subparameter appears as the last subparameter in the VOLUME parameter. Follow SER= with the volume serial numbers. The serial numbers must be enclosed in parentheses, unless there is only one serial number. If SER is the only subparameter you are coding, you can code VOLUME=SER=(serial number,...) or VOLUME=SER=serial number.

Do not use SCRATCH as a volume serial number because it is used to notify the operator to mount a nonspecific volume. For Optical Readers, if no volume serial number is specified, VOLUME=SER=OCRINP is assumed.

9. When you have the same volume serial number on more than one DD statement within a job step, the UNIT parameter will be checked only on the first DD statement.
10. **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.

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.

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.

DD

Examples of the VOLUME Parameter

```
//DD1 DD DSNAME=STEP,UNIT=2314,DISP=OLD X  
// VOLUME=(PRIVATE,,,SER=548863)
```

This DD statement defines an existing data set and informs the system that the data set resides on the volume whose serial number is 548863. Since PRIVATE is coded in the VOLUME parameter, the system will not assign the volume to any data set for which a nonspecific volume request is made and will cause the volume to be demounted after its use in the job step.

```
//DDB DD DSNAME=COMM,DISP=(NEW,KEEP),SPACE=(CYL,(30,2)), X  
// VOLUME=(PRIVATE,,,2),UNIT=2314
```

The DD statement named DDB defines a new data set for which the system is to assign a volume. Since only one device is requested (UNIT=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), X  
// VOLUME=(,,,4,SER=(96341,96342))
```

This DD statement defines an existing data set, which resides on the volumes whose serial numbers are 96341 and 96342, and requests that a total of 4 volumes be used to process the data set if required.

```
//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 can satisfy a request for a specific volume that is already mounted:

1. The volume is permanently resident or reserved. The use attribute of the volume does not affect assignment of the volume and the use attribute is not changed.
2. The direct access volume is a removable volume that has not been assigned the nonsharable attribute and is being used by a concurrently executing step. (If your request would make the volume nonsharable, the system waits to assign you that volume until all other job steps using the volume have terminated.) The volume remains private if its use attribute is private. The volume becomes private if the use attribute is public and the request is for a private volume. The volume remains public if its use attribute is public and the request is for a public volume.
3. The direct access volume is a removable public volume and is not in use. The use attribute (private or public) assigned to the volume when it is allocated is determined by the presence or absence of the PRIVATE subparameter.
4. The tape volume is a scratch volume and is not in use. The use attribute of private is assigned to the volume.

Satisfying Nonspecific Volume Requests

There are four types of nonspecific volume requests that can be made:

1. You can request a private volume for a temporary data set.
2. You can request a private volume for a nontemporary data set.
3. You can request a public volume for a temporary data set.
4. You can request a storage volume for a nontemporary data set.

How the system satisfies these different types of requests are described below. Since the system satisfied the first two types of requests in the same way, these two requests are described together.

1. When you make a nonspecific volume request for a private direct access or tape volume, the system assigns a volume that is mounted but not in use, or requests the operator to mount a volume. The operator should mount a volume whose space is unused. This allows you to have control over all space on the volume. Once mounted, the volume is assigned the use attribute of private.
2. When you make a nonspecific volume request for a public direct access volume that is to contain a temporary data set, the system assigns a public or storage volume that is already mounted, or requests the operator to mount a removable volume. If a mounted volume is selected, its use attribute is not affected. If a removable volume is mounted, it is assigned the use attribute of public.

When you make a nonspecific volume request for a public **tape** volume that is to contain a temporary data set, the system assigns a scratch volume that is already mounted, or it requests the operator to mount a tape volume. Once mounted, the volume is assigned the use attribute of scratch.

3. When you make a nonspecific volume request for a public **direct access** volume that is to contain a nontemporary data set, the system assigns a storage volume if one is mounted. Otherwise, the request is treated as a nonspecific volume request for a private volume.

When you make a nonspecific volume request for a public **tape** volume that is to contain a nontemporary data set, the request is treated as a nonspecific volume request for a private volume.

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* or *Operator's Library: OS/VS2 References, GC38-0210*.

```
┌ // command operand comments
```

The command statement consists of the characters `//` in columns 1 and 2, and three fields -- the operation (command), operand, and comments fields.

The following commands can be entered through the input stream.

CANCEL: The CANCEL command 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.

DISPLAY: The DISPLAY command causes a console display of certain system status information.

HOLD: The HOLD command temporarily prevents one job or all jobs from being selected for processing.

LOG: The LOG command enters information into the system log.

MODIFY: The MODIFY command changes the characteristics of a functioning output writer.

MOUNT: The MOUNT command 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.

RELEASE: The RELEASE command enables the system to resume job selection, which had been suspended by the HOLD command or `TYPRUN=HOLD` on the JOB statement.

REPLY: The REPLY command is used to reply to messages from the system or from a processing program that requests information.

RESET: The RESET command changes the class or priority of a job in an input, hold, or system output queue.

SET: The SET command establishes the values of certain variables, such as the time of day and the date.

START: The START command starts a particular system process, e.g., an input reader, graphic job processor, initiator, etc.

STOP: The STOP command stops a system process that had been previously started by a START command stops the console display effected by the DISPLAY command.

UNLOAD: The UNLOAD command removes the volume previously mounted in response to a MOUNT command.

VARY: The VARY command places an I/O device or path into an online or offline status.

WRITELOG: The WRITELOG command has the system output writer write out the contents of the system log.



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 and it contains errors, the command will not be executed. Furthermore, you will receive no indication that the command was not executed.
9. If you include a command statement as part of your job control statements, the command will usually be executed as soon as it is read. Because of this, it is not likely that the command will be synchronized with the execution of the job step to which it pertains. Therefore, you should preferably tell the operator which commands you want issued and when they should preferably be issued, and let him issue them.
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 tells the system to start an initiator. The characters A and B indicate that the initiator is to select for execution only jobs from classes A and B.

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 will be able to 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.
```

Comme

The Delimiter Statement

Control Statement

The delimiter statement indicates the end of data submitted through an input stream for a step.

```
/*      comments
```

The delimiter statement consists of the characters /* in columns 1 and 2 and the comments field.

Rules for Coding

1. The system will recognize 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 need not 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
```

Delimit

The Null Statement

Control Statement

The null statement indicates to the system that the job just read is to 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 will place 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 will recognize it as the last job in the input stream and place it on the queue.
4. The system will flush any control statements or data between a null statement and the next JOB statement.
5. If a null statement follows a control statement that is being continued, the system treats the null statement as a blank comment field and assumes that the control statement contains no other operands.

Example of the Null Statement

```
//MYJB      JOB      , 'C BROWN'  
//STEP1    EXEC     PROC=FIELD  
//STEP2    EXEC     PGM=XTRA  
//DD1      DD       UNIT=2400  
//DD2      DD       *  
  
.  
.  
data  
.  
.  
.  
  
/*  
//
```

Null

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 four fields -- the name field, the operation (PEND) field, and the comments field.

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 FOR INSTREAM PROCEDURES
```

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.



PEND

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.

```
┌ //name      PROC      operands      comments
```

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

Rules for Coding

1. A PROC statement is required for an in-stream procedure; 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. A name is required for in-stream procedures.
3. If you code a name, follow it with one or more blanks. 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.
In an in-stream procedure, the operand field is required.
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 (the enclosing apostrophes 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 will be assigned.
11. If you concatenate the symbolic parameter with some other information, (e.g., &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,      X
//          VOLUME=SER=888888 .
//DD2      DD        DSNAME=&LIBRARY,DISP=( OLD,KEEP ),UNIT=2314,      X
//          VOLUME=SER=&NUMBER
```

Three symbolic parameters are defined in this cataloged procedure: ϵ STATUS, ϵ LIBRARY, and ϵ NUMBER. Values are assigned to the symbolic parameters on the PROC statement. These values will be used when the procedure is called and values have not been assigned to the symbolic parameters by the programmer.

```
//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, i.e., DD * or DD DATA statements). It is used in job steps that call procedures to postpone defining data in the input stream on an overriding DD statement until the last overriding DD statement for a procedure step. (Overriding DD statements must appear in the same order as the corresponding DD statements in the procedure.)

When You Code the DDNAME Parameter

When the system encounters a DD statement that contains the DDNAME parameter, it saves the ddname of that statement. The system also temporarily saves the name specified in the DDNAME parameter so that it can relate that name to the ddname of a later DD statement. Once a DD statement with that corresponding name is encountered, the name is no longer saved. For example, if the system encounters this statement

```
//XYZ DD DDNAME=PHOB
```

the system saves XYZ and, temporarily, PHOB. Until the ddname is encountered in the input stream, the data set is a dummy data set.

When the system encounters a statement whose ddname has been temporarily saved, it does two things. It uses the information contained on this statement to define the data set; it associates this information with the name of the statement that contained the DDNAME parameter. The value that appeared in the DDNAME parameter is no longer saved by the system. To continue the above example, if the system encounters this statement

```
//PHOB DD DSN= 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 DSN= 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,           X
//      VOLUME=SER=46231,SPACE=(TRK,(20,5))
      .
      .
//DD12  DD   DSN=SET13,DISP=(NEW,KEEP),VOLUME=REF=*.DD1,     X
//      SPACE=(TRK,(40,5))
```

When you want to concatenate data sets, the unnamed DD statements must follow the DD statement that contains the DDNAME parameter, not the DD statement that defines the data set. The following sequence of control statements illustrates this:

```
//DDA    DD   DDNAME=DEFINE
//      DD   DSN=A.B.C,DISP=OLD
//      DD   DSN=SEVC,DISP=OLD,UNIT=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.

The DCB Subparameters BLKSIZE and BUFNO

Two DCB subparameters can be coded with the DDNAME parameter -- BLKSIZE and BUFNO. This allows you to assign these DCB characteristics to the data set defined in the referenced DD statement. When the DCB subparameters BLKSIZE and BUFNO are coded both on the DD statement that contains the DDNAME parameter and on the referenced DD statement, the subparameters coded on the former are ignored.

These subparameters would most often be coded with the DDNAME parameter when the referenced DD statement defines data in the input stream. Data in the input stream is written on a direct access device, and the records are blocked as they are written. The input reader procedure normally assigns a block size and number of buffers for blocking. Coding the BLKSIZE subparameter allows you to specify that you want shorter blocks. Coding the BUFNO subparameter allows you to specify that you want fewer buffers. You cannot specify that you want larger blocks or more buffers than would be assigned by the input reader procedure. (When a job is submitted via remote job entry and the BUFNO subparameter is coded, the BUFNO subparameter is ignored.)

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 is assigned a disposition of KEEP or CATLG. (A data set assigned a disposition of KEEP may be assigned a disposition of CATLG by a later job step or job.) The name you assign to a nontemporary data set must be specified in the DSNAME parameter by all other steps and jobs that want to use the data set.

A nontemporary data set name can be either an unqualified or qualified name. An unqualified data set name consists of 1 through 8 characters. The first character must be an alphabetic or national (@,#,\$) character; the remaining characters can be any alphabetic or national characters, a hyphen, or a plus zero (12-0) punch).

A qualified data set name consists of 1 through 44 characters (including periods), except when the qualified name identifies a generation data group. In this case, the data set name may consist of only 1 through 35 characters (including periods). For each eight characters or less there must be a period, and the first character of the name and the character following a period must be an alphabetic or national (@,#,\$) character.

If you assign a qualified name to a data set that is to be cataloged all but the lowest level of the name must already exist as indexes in the system catalog before you can request the system to catalog the data set. An index level is created by using the IEHPROGM utility program. Once the indexes are established, the data set can be cataloged.

When you request a data set that is cataloged on a control volume other than the system catalog, the system attempts to mount this control volume if it is not already mounted. After the system obtains the pointer to this data set, the control volume may then be demounted by the system if the unit on which it was mounted is required by another volume. If you plan to delete, uncatalog, or recatalog the data set, the volume must be mounted during disposition processing (at the end of the job step) in order for the pointer to be deleted or revised. You can ensure that the volume remains mounted by requesting the operator to issue a MOUNT command for this volume before the job step is initiated. If you do not use the MOUNT command to mount the volume and if the volume is not mounted during disposition processing,

then, after the job has terminated, use the IEHPROGM utility program to delete or revise the pointer in the control volume. (In order for the system to mount a control volume, the control volume must be logically connected to the system catalog. This is done using the CONNECT function of the IEHPROGM utility program, which is described in the Utilities publication.)

Members of a Partitioned Data Set

A partitioned data set consists of independent groups of sequential records, each identified by a member name in a directory. When you want to add a member to a partitioned data set or retrieve a member, you specify the partitioned data set name and follow it with the member name. The member name is enclosed in parentheses and consists of 1 to 8 characters. The first character must be an alphabetic or national (@,\$,#) character; the remaining characters can be any alphanumeric or national characters.

Generations of 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, you specify the generation data group name and follow it with the generation number. The generation number is enclosed in parentheses and the number is a zero or a signed integer. A zero represents the most current generation of the group; a negative integer (e.g., -1) represents an older generation; a positive integer (e.g., +1) represents a new generation that has not as 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 Appendix E in this publication.

Areas of an Indexed Sequential Data Set

The areas used for an indexed sequential data set are the index, prime, and overflow areas. When you are creating the data set and define any of these areas on a DD statement, you must identify the data set name and follow it with the area name you are defining. The area name is enclosed in parentheses and is either PRIME, INDEX, or OVFLOW. If you are using only one DD statement to define the entire data set, code DSNNAME=dsname or DSNNAME=dsname(PRIME). When you retrieve the data set, you code only the data set name; you 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 DSNNAME=εεdsname.
- A member of a temporary partitioned data set by coding DSNNAME=εεdsname(member name).
- An area of a temporary indexed sequential data set by coding DSNNAME=εεdsname(area name).

Temporary Data Sets

Any data set that is created and deleted within the same job is a temporary data set. A DD statement that defines a temporary data set need not include the DSNNAME parameter; the system generates one for you.

If you do include the DSNNAME parameter, the temporary data set name can consist of 1 through 8 characters and is preceded by two ampersands (εε). The character following the ampersands must be an alphabetic or national (@,#,\$) characters; 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, which begins with SYS and includes the jobname, the temporary name assigned in the DSNNAME parameter, and other identifying characters.

If you attempt to keep or catalog a temporary data set (you specify a disposition of KEEP or CATLG in the DISP parameter), the system changes the disposition to PASS and the data set is deleted at job termination. However, this change is not made for a data set on a tape volume when the following conditions exist: (1) the data set is new; (2) the data set is not assigned a name; and (3) DEFER is specified in the UNIT parameter. The data set is deleted at job termination, but the system tells the operator to keep the volume on which the data set resided during the job.

Members of a Temporary Partitioned Data Set

When you want to add a member to a temporary partitioned data set or retrieve a member during the job, you specify the partitioned data set's temporary name and follow it with the member name. The member name is enclosed in parentheses and consists of 1 through 8 characters. The first character must be an alphabetic or national (@,\$,#) character; the remaining characters can be any alphameric or national characters.

Areas of a Temporary Indexed Sequential Data Set

The areas used for an indexed sequential data set are the index, prime, and overflow areas. When you are creating a temporary indexed sequential data set and define any of these areas on a DD statement, you must identify the data set's temporary name and follow it with the area name you are defining. The area name is enclosed in parentheses and is either PRIME, INDEX, or OVFLOW. If you are using only one DD statement to define the entire temporary data set, code DSNNAME=εεdsname or DSNNAME=εεdsname(PRIME). If you want to retrieve the temporary data set on the same job, you code only the data set's temporary name; you do not include the term PRIME, INDEX, or OVFLOW.

Using a Dedicated Data Set

If your installation provides dedicated data sets, you can use these data sets to contain your data instead of creating your own temporary data sets. The use of dedicated data sets eliminates some of the time required to schedule a job step since the data sets are already allocated.

To use a dedicated data set, code `DSNAME=εεname` or `DSNAME=εname` on a DD statement, along with all other parameters required to define your temporary data set, e.g., `UNIT`, `SPACE`, `DCB`. Replace the term "name" with the ddname of the DD statement in the initiator cataloged procedure that defines the dedicated data set you want to use. If the system cannot assign you this dedicated data set, the parameters coded on your DD statement are used to create a temporary data set.

Copying the Data Set Name From an Earlier DD Statements

The name of a data set that is used several times in a job, whether specified in the `DSNAME` parameter or assigned by the system, can be copied after its first use in the job. This allows you to easily change data sets from job to job and eliminates your having to assign names to temporary data sets. To copy a data set name, refer to an earlier DD statement that identifies the data set. When the earlier DD statement is contained in an earlier job step, you code `DSNAME=*.stepname.ddname`; when the earlier DD statement is contained in the same job step, you code `DSNAME=*.ddname`; when the earlier DD statement is contained in a cataloged procedure step called by an earlier job step, you code `DSNAME=*.stepname.procstepname.ddname`.

Specifying the DSNAME Parameter in Apostrophes

Sometimes, it may be necessary or desirable to specify a data set name that contains special characters. If the name contains special characters, you must enclose the name in apostrophes (5-8 punch), e.g., `DSNAME='DAT+5'`. If one of the special characters is an apostrophe, you must identify it by coding two consecutive apostrophes (two 5-8 punches) in its place. e.g., `DSNAME='DAYSEND'`. A data set name enclosed in apostrophes can consist of 1 through 44 characters.

There are cases when your data set name must contain required special characters, which tell the system something about the data set (e.g., `εε` in `DSNAME=εεname` are required special characters that tell the system that this is a temporary data set). In these cases, the data set name must **not** be enclosed in apostrophes because the system will not recognize the required special characters as having any special significance. The following data set names contain special characters that tell the system something about the data set and, therefore, **cannot** be enclosed in apostrophes:

- `DSNAME=name(member name)`
- `DSNAME=name(area name)`
- `DSNAME=name(generation number)`
- `DSNAME=εεname`
- `DSNAME=*.stepname.ddname`

Keep the following rules in mind:

1. If the data set is to be cataloged, the data set name cannot be enclosed in apostrophes.
2. If the data set name begins with a blank character, the data set is assigned a temporary data set name by the system.
3. If the data set name ends with a blank characters, the blank is ignored.
4. If the only special character is a period or a hyphen, you need not enclose the data set name in apostrophes.

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).

The Data Set Sequence Number Subparameter

When you want to place a data set on a tape volume that already contains one or more data sets, you must specify where the data set is to be placed, i.e., the data set is to be the second, third, fourth, etc., data set on the volume. The data set sequence number causes the tape to be positioned properly so that the data set can be written on the tape or retrieved.

The data set sequence number subparameter is a positional subparameter and is the first subparameter that can be coded. The data set sequence number is a 1- to 4-digit number. The system assumes 1, i.e., this is the first data set on the reel, if you omit this subparameter or if you code 0, unless the data set is a passed or cataloged data set. If a data set is cataloged, the system obtains the data set sequence number from the catalog; for a passed data set, the data set sequence number is obtained from the passing step.

When you request the system to bypass label processing (BLP is coded as the label type in the LABEL parameter) and the tape volume contains labels, the system treats anything between tapemarks as a data set. Therefore, in order for the tape with labels to be positioned properly, the data set sequence number must reflect all labels and data sets that precede the desired set. Section I of the **Tape Labels** publication illustrates where tapemarks appear.

The Label Type Subparameter

The label type subparameter tells the system what type of labels 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 tapemarks as a data set. Therefore, in order for a tape with labels to be positioned properly, the data set sequence number subparameter of the LABEL parameter must be coded and the subparameter must reflect all labels and data sets that precede the desired data set. Section I of the **Tape Labels** publication illustrates where tapemarks appear.

The label type subparameter can also be specified when you make a nonspecific volume request for a tape volume (i.e., no volume serial numbers are specified on the DD statement) and you want the data set to have a certain type of labels. If the volume that is mounted does not have the corresponding label type you desire, you may be able to change the label type.

When you specify NL or NSL and the operator mounts a tape volume that contains standard labels, you may use the volume provided: (1) the expiration date of the existing data set on the volume has passed; (2) the existing data set on the volume is not password protected; and (3) you make a nonspecific volume request. All of these conditions must be met. If they are not, the system requests the operator to mount another tape volume.

If you specify SL and make a nonspecific volume request, but the operator mounts a tape volume that contains other than IBM standard labels, the system requests the operator to identify the volume serial number and the volume's new owner before the IBM standard labels are written. If the tape volume has American National Standard labels, the system asks the operator for permission to destroy the label. If you specify SL and make a specific volume request, but the volume that is mounted does not contain IBM standard labels, the system rejects the tape and requests the operator to mount the tape volume specified.

The PASSWORD and NOPWREAD 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.

The IN and OUT Subparameters

The basic sequential access method (BSAM) permits a specification of INOUT or OUTIN in the OPEN macro instruction as the processing method. If you have specified either of these processing methods in the OPEN macro instruction and want to override it, you may be able to do so by coding either the IN or OUT subparameter. For FORTRAN users, the IN and OUT subparameters provide a means of specifying how the data set is to be processed, i.e., for input or output.

When INOUT is specified in the OPEN macro instruction and you want the data set processed for input only, you can specify the IN subparameter. When the IN subparameter is coded, any attempt by the processing program to process the data set for output is treated as an error.

When OUTIN is specified in the OPEN macro instruction and you want the data set processed for output only, you can specify the OUT subparameter. When the OUT subparameter is coded, any attempt by the processing program to process the data set for input is treated as an error.

The IN and OUT subparameters are positional subparameters. If either is coded, it must appear as the fourth subparameter, after the data set sequence number subparameter, the label type subparameter, and the PASSWORD subparameter, or the commas that indicate their absence.

The RETPD and EXPDT Subparameters

When it is necessary that a data set be kept for some period of time you can tell the system how long it is to be kept when you create the data set. As long as the time period has not expired, a data set that resides on a direct access volume cannot be deleted by or overwritten by another job step or job. (If it is necessary to delete a data set, you can use the IEHPROGM utility program to delete the data set. The IEHPROGM utility program is described in the *Utilities* publication.)

There are two different ways to specify a time period: (1) tell the system how many days you want the data set kept, the RETPD subparameter, or (2) tell the system the exact date after which the data set need not be kept, the EXPDT subparameter.

If you code the RETPD subparameter, you specify a 1- to 4-digit number, which represents the number of days the data set is to be kept. If you code the EXPDT subparameter, you specify a 2-digit year number and a 3-digit day number (e.g., January 1 would be 001, July 1 would be 182), which represents the date after which the data set need 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.

This page replaces the following sections of the base document, **OS/VS JCL Reference, GC28-0618-1**.

- **Appendix B: Writing Cataloged and In-stream Procedures, pages 193-196.**

The information formerly contained in Appendix B can now be found in the chapter entitled "Cataloged and In-stream Procedures" in **OS/VS JCL Services, GC28-0617-2**.

- **Appendix C: Requesting Space for a Data Set on a Direct Access Volume, pages 197-204.**

The information formerly contained in Appendix C can now be found in the chapters entitled "Requesting Space for a Single Data Set" and "Requesting Space for a Group of Data Sets" in **OS/VS JCL Services, GC28-0617-2**.

Appendix D: Creating and Retrieving Indexed Sequential Data Sets

Indexed sequential (ISAM) data sets are created and retrieved using special subsets of DD statement parameters and subparameters. Each data set can occupy up to three different areas of space:

1. Prime area -- This area contains data and related track indexes. It exists for all indexed sequential data sets.
2. Overflow area -- This area contains overflow from the prime area when new data is added. It is optional.
3. Index area -- This area contains master and cylinder indexes associated with the data set. It exists for any indexed sequential data set that has a prime area occupying more than one cylinder.

Indexed sequential data sets must reside on direct access volumes. The data set can reside on more than one volume and the device types of the volumes may in some cases differ.

Creating an Indexed Sequential Data Set

One to three DD statements can be used to define a new indexed sequential data set. When you use three DD statements to define the data set, each DD statement defines a different area and the areas must be defined in the following order:

1. Index area.
2. Prime area.
3. Overflow area.

When you use two DD statements to define the data set, the areas must be defined in the following order:

- | | | |
|----------------|----|-------------------|
| 1. Index area. | or | 1. Prime area. |
| 2. Prime area. | | 2. Overflow area. |

When you use one DD statement to define the data set, you are defining the prime area and, optionally, the index area.

When more than one DD statement is used to define the data set, assign a ddname only to the first DD statement; the name field of the other statements must be blank.

The only DD statement parameters that can be coded when defining a new indexed sequential data set are the DSNAME, UNIT, VOLUME, LABEL, DCB, DISP, SPACE, SEP, and AFF parameters. When to code each of these parameters and what restrictions apply are described in the following paragraphs.

The DSNAME Parameter

The DSNAME parameter is required on any DD statement that defines a new temporary or nontemporary indexed sequential data set. To identify the area you are defining, you follow the DSNAME parameter with the area: DSNAME=name(INDEX), DSNAME=name(PRIME), or DSNAME=name(OVFLOW). If you are using only one DD statement to define the data set, code DSNAME=name(PRIME) or DSNAME=name.

When reusing previously allocated space to create an ISAM data set, the DSNAME parameter must contain the name of the old data set to be overlaid.

The UNIT Parameter

The UNIT parameter is required on any DD statement that defines a new indexed sequential data set unless VOLUME=REF=reference is coded. You must request a direct access device in the UNIT parameter and must **not** request DEFER.

If there are separate DD statements defining the prime and index areas, you must request the same number of direct access devices for the **prime** area as there are volumes specified in the VOLUME parameter. You may request only one direct access volume for an index area and one for an overflow area.

A DD statement for the index area or overflow area can request a device type different than the type requested on the other statements.

Another way to request a device is to code UNIT=AFF=ddname, where the named DD statement requests the direct access device or device type you want.

The VOLUME Parameter

The VOLUME parameter is required only if you want an area of the data set written on a specific volume or the prime area requires use of more than one volume. (If the prime area and index area are defined on the same statement, you cannot request more than one volume on the DD statement.) Either supply the volume serial number or numbers in the VOLUME parameter or code VOLUME=REF=reference. In all cases, the VOLUME parameter can be used to request a private volume (PRIVATE) and to retain the private volume (RETAIN).

The LABEL Parameter

The LABEL parameter need only be coded to specify a retention period (EXPDT or RETPD) or password protection (PASSWORD).

The DCB Parameter

The DCB parameter must be coded on every DD statement that defines an indexed sequential data set. At minimum, the DCB parameter must contain DSORG=IS or DSORG=ISU. Other DCB subparameters can be coded to complete the data control block if it has not been completed by the processing program. When more than one DD statement is used to define the data set, code all the DCB subparameters on the first DD statement. Code DCB=*.ddname on the remaining statement or statements; ddname is the name of the DD statement that contains the DCB subparameters.

When reusing previously allocated space and recreating an ISAM data set, desired changes in the DCB parameter must be coded on the DD statement. Although you are creating a new data set, some DCB subparameters cannot be changed if you want to use the space the old data set used. The DCB subparameters you can change are: BFALN, BLKSIZE, CYLOFL, DSORG, KEYLEN, LRECL, NCP, NTM, OPTCD, RECFM, and RKP.

The DISP Parameter

If you are creating a new data set and not reusing preallocated space, the DISP parameter need only be coded if you want to keep, DISP=(,KEEP), catalog, DISP=(,CATLG), or pass, DISP=(,PASS), the data set. If you are reusing previously allocated space and recreating an ISAM data set, code DISP=OLD. The newly created data set will overlay the old one.

In order to catalog the data set when DISP=(,CATLG) is coded or pass the data set when DISP=(,PASS) is coded, the data set must be defined on only one DD statement. If the data

set was defined on more than one DD statement and the volumes on which the data set now resides correspond to the same device type, you can use the IEHPROGM utility program to catalog the data set. Refer to the chapter "The IEHPROGM Program" in the Utilities publication for details.

The SPACE Parameter

The SPACE parameter is required on any DD statement that defines a new indexed sequential data set. Use either the recommended nonspecific allocation technique or the more restricted absolute track (ABSTR) technique. If more than one DD statement is used to define the data set, all must request space using the same technique.

Nonspecific Allocation Technique

You must request the primary quantity in cylinders (CYL). When the DD statement that defines the prime area requests more than one volume, each volume is assigned the number of cylinders requested in the SPACE parameter.

One of the subparameters of the SPACE parameter, the "index" subparameter, is used to indicate how many cylinders are required for an index. When one DD statement is used to define the prime and index areas and you want to explicitly state the size of the index, code the "index" subparameter.

The CONTIG subparameter can be coded in the SPACE parameter. However, if CONTIG is coded on one of the statements, it must be coded on all of them.

You cannot request a secondary quantity for an indexed sequential data set. Also, you cannot code the subparameters RLSE, MXIG, ALX, and ROUND.

Absolute Track Technique

The number of tracks you request must be equal to one or more whole cylinders. The address of the beginning track must correspond with the first track of a cylinder other than the first cylinder on the volume. When the DD statement that defines the prime area requests more than one volume, space is allocated for the prime area beginning at the specified address and continuing through the volume and onto the next volume until the request is satisfied. (This can only be done if the volume table of contents of the second and all succeeding volumes is contained within the first cylinder of each volume.)

One of the subparameters of the SPACE parameter, the "index" subparameter, is used to indicate how many tracks are required for an index. The number of tracks specified must be equal to one or more cylinders. When one DD statement is used to define the prime and index areas and you want to explicitly state the size of the index, code the "index" subparameter.

The SEP or AFF Parameter

The SEP or AFF parameter is coded only if you want channel separation from the area or areas defined on the preceding statement or statements in the group. In order for the areas to be written using separate channels, you must also request devices by their actual address, e.g., UNIT=190.

Note: If the indexed sequential data set is to reside on more than one volume and an error is encountered as the volumes are being allocated to the data set, follow this procedure before resubmitting the job: Use the IEHPROGM utility program to scratch the data set labels on any of the volumes to which the data set was successfully allocated. This utility program is described in the chapter "The IEHPROGM Program" in the Utilities publication.

Area Arrangement of an Indexed Sequential Data Set

When you create an indexed sequential data set, the arrangement of the areas is based on two criteria:

1. The number of DD statements used to define the data set.
2. What area each DD statement defines.

An additional criterion is used when you do not include a DD statement that defines the index area:

3. Is an index size coded in the SPACE parameter on the DD statement that defines the prime area?

Figure 7 in Appendix F illustrates the different arrangements that can result based on the criteria listed above. In addition, Figure 7 indicates what restrictions apply on the number and types of devices that can be requested.

Retrieving an Indexed Sequential Data Set

If all areas of an existing indexed sequential data set reside on volumes of the same device type, you can retrieve the entire data set with one DD statement. If the index or overflow resides on a volume of a different device type, you must use two DD statements. If the index and overflow reside on volumes of different device types, you must use three DD statements to retrieve the data set. The DD statements are coded in the following order:

1. First DD statement - defines the index area
2. Second DD statement - defines the prime area
3. Third DD statement - defines the overflow area

The only DD statement parameters that can be coded when retrieving an indexed sequential data set are the DSNAME, UNIT, VOLUME, DCB, and DISP parameters. When to code each of these parameters and what restrictions apply are described in the following paragraphs.

The DSNAME Parameter

The DSNAME parameter is always required. Identify the data set by its name, but do not include the term INDEX, PRIME, or OVFLOW. If the data set was passed from a previous step, identify it by a backward reference.

The UNIT Parameter

The UNIT parameter must be coded unless the data set resides on one volume and was passed. You identify in the UNIT parameter the device type and how many of these devices are required.

If the data set resides on more than one volume and the volumes correspond to the same device type, you need only one DD statement to retrieve the data set. Request one device in the UNIT parameter per volume. If the index or overflow area of the data set resides on a different type of volume than the other areas, you must use two DD statements to retrieve the data set. On one DD statement, request the device type required to retrieve the index or overflow area. On the other DD statement, request the device type and the number of devices required to retrieve the prime area and the overflow area if the overflow area resides on the same device type. If the index and the overflow areas reside on different device types from the prime area, a third DD statement is needed.

The VOLUME Parameter

The VOLUME parameter must be coded unless the data set resides on one volume and was passed from a previous step. Identify in the VOLUME parameter the serial numbers of the volumes on which the data set resides. Code the serial numbers in the same order as they were coded on the DD statements used to create the data set.

The DCB Parameter

The DCB parameter must be coded unless the data set was passed from a previous step. The DCB parameter must always contain DSORG=IS or DSORG=ISU. Other DCB subparameters can be coded to complete the data control block if it has not been completed by the processing program.

The DISP Parameter

The DISP parameter must always be coded. The first subparameter of the DISP parameter must be MOD or OLD. You can, optionally, assign a disposition as the second subparameter.

Examples of Creating and Retrieving an Indexed Sequential Data Set

The following job step includes the DD statement that could be used to create an indexed sequential data set. Each area of the indexed sequential data set is defined on a separate DD statement.

```
//OUTPUT4 EXEC PGM=INCLUDE
//GROUP1 DD DSN=PART86(INDEX),DISP=(,KEEP),UNIT=2314 X
// VOLUME=SER=538762,SPACE=(CYL,10,,CONTIG), X
// DCB=(DSORG=IS,RECFM=F,LRECL=80,RKP=1,KEYLEN=8)
// DD DSN=PART86(PRIME),DISP=(,KEEP),UNIT=(2314,2),X
// VOLUME=SER=(538763,538764), X
// SPACE=(CYL,(25),,CONTIG),DCB=*.GROUP1
// DD DSN=PART86(OVFLOW),DISP=(,KEEP),UNIT=2314, X
// VOLUME=SER=538765,SPACE=(CYL,15,,CONTIG), X
// DCB=*.GROUP1
```

The following job step includes the DD statements required to retrieve the indexed sequential data set created above.

```
//INPUT12 EXEC PGM=ADD
//RET4 DD DSN=PART86,DCB=DSORG=IS,UNIT=2314, X
// DISP=OLD,VOLUME=SER=538762
// DD DSN=PART86.DCB=DSORG=IS,UNIT=(2314,3), X
// DISP=OLD,VOLUME=SER=(538763,538764,838765)
```

Two DD statements are required to retrieve the data set because the index area resides on a volume of a different device type than the volumes on which the prime and overflow areas reside.



Appendix E: Creating and Retrieving Generation Data Sets

A generation data set is one of a collection of successive, historically related, cataloged data sets known as a generation data group. The system keeps track of each data set in a generation data group as it is created so that new data sets can be chronologically ordered and old ones easily retrieved.

To create or retrieve a generation data set, you identify the generation data group name in the DSNAME parameter and follow the group name with a relative generation number. When creating a generation data set, the relative generation number tells the system whether this is the first data set being added during the job, the second, the third, etc. When retrieving a generation data set, the relative generation number tells the system how many data sets have been added to the group since this data set was added.

A generation data group can consist of cataloged sequential, partitioned, indexed sequential (if the data set is defined on one DD statement), and direct data sets residing on tape volumes, direct access volumes, or both. Generation data sets can have like or unlike DCB attributes and data set organizations. If the attributes and organizations of all generations in a group are identical, the generations can be retrieved together as a single data set (up to 255 data sets can be retrieved in this way).

Before You Define the First Generation Data Set

Before you define the first generation data set, you must build a generation data group index. This index provides lower-level entries for as many generation data sets (up to 255) as you would like to have in your generation data group. The system uses these lower-level indexes to keep track of the chronological order of the generation data sets. The index must reside on the system residence volume, or an alternate control volume. You use the IEHPROGM utility program to build your index; this program is described in the chapter "The IEHPROGM Program" in the Utilities publication.

Another requirement of generation data groups is that a data set label list exist on the same volume as the index. The system uses this label to refer to DCB attributes when you define a new generation data set. There are two ways to satisfy this requirement: (1) create a model data set label before you define the first generation data set; or (2) use the DCB parameter to refer the system to an existing cataloged data set each time you define a new generation data set.

Creating a Model Data Set Label

To create a model data set label, you must define a data set and request that it be placed on the same volume as the generation data group index. This ensures that there is always a data set label on the same volume as the index to which the system can refer.

The name you assign to the data set may be the same or different than the name assigned to the generation data group. (If you assign the same name for both, the data set associated with the model data set label cannot be cataloged.) You may request a space allocation of zero tracks or cylinders. The DCB attributes you can supply are DSORG, OPTCD, BLKSIZE, LRECL, KEYLEN, and RKP.

This is an example of creating a model data set label:

```
//DD1 DD DSNAME=PAY.WEEK,DISP=(NEW,KEEP),UNIT=2314, X
// VOLUME=SER=SYSRES,SPACE=(TRK,0),DCB=(RECFM=FB, X
// LRECL=240,BLKSIZE=960)
```

You need not create a model data set label for every generation data group whose indexes reside on the same volume. Instead, you may create one model data set label to be used by any number of generation data groups. If you create only one model, you should not supply any DCB attributes. When you create a generation data set, you specify the name of the model in the DCB parameter and follow the name with a list of all the DCB subparameters required for the new generation data set, i.e., DCB=(dsname,list of attributes).

Referring the System to a Cataloged Data Set

If there is a cataloged data set that resides on the same volume as your generation data group index and you are sure that data set will exist as long as you are adding data sets to your generation data group, you need not create a model data set label. When you create a generation data set, you specify the name of the cataloged data set in the DCB parameter, i.e., DCB=dsname. If all the DCB attributes are not contained in the label of the cataloged data set, or if you want to override certain attributes, follow the data set name with these attributes, i.e., DCB=(dsname,list of attributes).

Creating a Generation Data Set

When defining a new generation data set, you always code the DSNAMES, DISP, and UNIT parameters. Other parameters you might code are the VOLUME, SPACE, LABEL, and DCB parameters.

The DSNAMES Parameter

In the DSNAMES parameter, you code the name of the generation data group followed by a number enclosed in parentheses. This number must be 1 or greater. If this is the first data set you are adding to a particular generation data group during the job, code +1 in parentheses. Each time during the job you add a data set to the same generation data group, increase the number by one.

Any time you refer to this data set later in the job, you use the same relative generation number as was used earlier. At the end of the job, the system updates the relative generation numbers of all generations in the group to reflect the additions.

The DISP Parameter

New generations are assigned a status of NEW and a disposition of CATLG in the DISP parameter, i.e., DISP=(NEW,CATLG). If you do not specify a disposition, or specify a disposition other than CATLG, the system assumes CATLG.

The UNIT Parameter

The UNIT parameter is required on any DD statement that defines a new generation data set unless VOLUME=REF=reference is coded. In the UNIT parameter, you identify the type of devices you want (tape or direct access).

Another way to request a device is to code UNIT=AFF=ddname, where the named DD statement requests the device or device type you want.

The VOLUME Parameter

You may assign a volume in the VOLUME parameter or let the system assign one for you. The VOLUME parameter can also be used to request a private volume (PRIVATE), to retain the private volume (RETAIN), and to indicate that more volumes may be required (volume count).

The SPACE Parameter

The SPACE parameter is coded only when the generation data set is to reside on a direct access volume. The SPLIT or SUBALLOC parameter can be coded in place of the SPACE parameter if the data set's organization permits the use of these parameters.

The LABEL Parameter

You can specify label type, password protection (PASSWORD), and a retention period (EXPDT or RETPD) in the LABEL parameter. If the data set will reside on a tape volume and is not the first data set on the volume, specify a data set sequence number.

The DCB Parameter

A model data set label that has the same name as the group name may exist. If this is so, and if the label contains all the attributes required to define this generation, you need not code the DCB parameter. If all the attributes are not contained in the label, or if you want to override certain attributes, code these attributes in the DCB parameter, i.e., DCB=(list of attributes).

If a model data set label has a different name than the group name and if the label contains all the attributes required to define this generation data set, only the name of the data set associated with the model data set label need be coded. Code the name in the DCB parameter, i.e., DCB=dsname. If all the attributes are not contained in the label, or if you want to override certain attributes, follow the data set name with these attributes, i.e., DCB=(dsname,list of attributes).

If a model data set label does not exist, you must code the name of a cataloged data set that resides on the same volume as the generation data group index, i.e., DCB=dsname. If all the attributes are not contained in the label for this data set, or if you want to override certain attributes, follow the data set name with these attributes, i.e., DCB=(dsname,list of attributes).

Retrieving a Generation Data Set

To retrieve a generation data set, you always code the DSNNAME and DISP parameters. Other parameters you might code are the UNIT, LABEL, and DCB parameters.

The DSNNAME Parameter

In the DSNNAME parameter, you code the name of the generation data group followed by a number enclosed in parentheses. The number you code depends on how many new generation data sets have been added to the group since this generation data set was added. If none have been added prior to the job, code a zero (0). If one has been added prior to the job, code (-1). Decrement the number by 1 until you determine the present relative generation number of the data set, then code this number.

Any time you refer to this data set later in the job, you use the same relative generation number as was used earlier, even if another generation has been added during the job.

If you want to retrieve all generations of a generation data group as a single data set, you specify the generation data group name without a generation number, e.g., `DSNAME=WEEKLY.PAYROLL`. You can retrieve all generations as a single data set only if the attributes and organization of all generations are identical.

If you want to retrieve all generations of a generation data group by concatenation, in order, starting with the most recent data set and with unity affinity to the most recent data set, code the generation data group name without a generation number, e.g., `DSNAME=WEEKLY.PAYROLL`. You can retrieve all generations by concatenating them only if the attributes and organization of all generation are identical.

The DISP Parameter

The DISP parameter must always be coded. The first subparameter of the DISP parameter must be OLD, SHR, or MOD. You can, optionally, assign a disposition as the second subparameter.

The UNIT Parameter

Code the UNIT parameter when you want more than one device assigned to the data set. Code the number of devices you want in the unit count subparameter, or, if the data set resides on more than one volume and you want as many devices as there are volumes, code P in place of the unit count subparameter.

The LABEL Parameter

Code the LABEL parameter when the data set has other than standard labels.

The DCB Parameter

Code the DCB parameter when the data set has other than standard labels and DCB information is required to complete the data control block.

Resubmitting a Job for Restart

Certain rules apply when you refer to generation data sets in a job resubmitted for restart (the RESTART parameter is coded on the JOB statement).

For step restart: If step restart is performed, generation data sets that were created and cataloged in steps preceding the restart step must not be referred to in the restart step or in steps following the restart step by means of the same relative generation numbers that were used to create them. Instead, you must refer to a generation data set by means of its present relative generation number. For example, if the last generation data set created and cataloged was assigned a generation number of +2, it would be referred to a 0 in the restart step and in steps following the restart step. In this case, the generation data set assigned number of +1 would be referred to as -1.

For checkpoint restart: If generation data sets created in the restart step were kept instead of cataloged (i.e., `DISP=(NEW,CATLG,KEEP)` was coded), you can, during checkpoint restart, refer to these data sets and generation data sets created and cataloged in steps preceding the restart step by means of the same relative generation numbers that were used to create them.

Generation data sets can be created and retrieved using utility programs. How to do this is described in *Utilities* publication.

Examples of Creating and Retrieving Generation Data Sets

The following job step includes the DD statements that could be used to add three data sets to a generation data group.

```
//STEP1 EXEC      PGM=PROCESS
//DD1  DD         DSNAME=A.B.C(+1),DISP=(NEW,CATLG),UNIT=2400,      X
//          VOL=SER=13846,LABEL=(,SUL)
//DD2  DD         DSNAME=A.B.C(+2),DISP=(OLD,CATLG),UNIT=2311      X
//          VOL=SER=10311
//DD3  DD         DSNAME=A.B.C(+3),DISP=(NEW,CATLG),UNIT=2301,      X
//          VOL=SER=28929,SPACE=(480,(150,20)),
//          DCB=(LRECL=120,BLKSIZE=480)      X
```

The first two DD statements do not include the DCB parameter; therefore, a model data set label must exist on the same volume as the generation data group index and must have the same name as the generation data group (A.B.C). Since the DCB parameter is coded on the third DD statement, the attributes LRECL and BLKSIZE, along with the attributes included in the model data set label, are used.

The following job includes the DD statements required to retrieve the generation data sets defined above when no other data sets have been added to the generation data group.

```
//JWC   JOB       CLASS=B
//STEP1 EXEC      PGM=REPORT9
//DDA   DD        DSNAME=A.B.C(-2),DISP=OLD,LABEL=(,SUL)
//ddb   DD        DSNAME=A.B.C(-1),DISP=OLD
//DDC   DD        DSNAME=A.B.C(0),DISP=OLD
```


Appendix F: Reference Tables

The first section of this appendix summarizes the DD statement parameters requires to perform the following functions:

- Create a data set on an 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 an 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 on 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)
System Output Devices	Location of the Data Set	SYSOUT	Required. Specifies the output class
		UNIT	Optional
	Size of the Data Set	SPACE	Optional
	Special Processing Option	OUTLIM	Optional. Meaningful only for Systems that utilize the Systems Management Facilities Option
		COPIES	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	
		DUMMY	Optional
Direct Access Devices	Data Set Information	DSNAME (or DSN)	Required if the data set is to be cataloged or used by a later job
		DISP	Required if the data set is to be cataloged, used by a later step in this job, or used by another job
	Location of the Data Set	UNIT	Required unless you request (with the VOLUME parameter) the same volume used for an earlier data set in your job, or unless you use the SPLIT or SUBALLOC parameters to allocate space to this data set
		VOLUME (or VOL)	Required if you want a specific volume or multiple volumes. If you do not use this parameter your data set will be allocated on any suitable volume
		LABEL	Required if you want the data set to have both IBM standard and user labels
	Size of the Data Set	SPACE	One of these parameters is required. SPLIT can only be used for BSAM or QSAM data sets. SPACE must be used for ISAM data sets
		SPLIT	
		SUBALLOC	
	Data Attributes	DCB	Optional. Required for BDAM and ISAM data sets
	Special Processing Options	SEP	Either parameter can be used
		AFF	
		DUMMY	Optional
		DYNAM	Optional. Meaningful only for TSO

Figure 3. DD Parameters for Creating a Data Set

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 Processing Option	DLM	Optional
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 Option	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	
DUMMY		Optional	
DYNAM		Optional. Meaningful only for TSO	
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	
DUMMY		Optional	
DYNAM		Optional. Meaningful only for TSO	

Figure 4. 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 Option	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	
		DUMMY	Optional
	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		
	DUMMY	Optional	
	DYNAM	Optional. Meaningful only for TSO	

Figure 5. 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 indexed sequential (ISAM) data set. The DD statements are coded in the following 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 Indexed 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 6. 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.
<p>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.</p> <p>² If the unused portion of the index area is less than one cylinder, it is used as an overflow area.</p>				

Figure 7. Area Arrangement of Indexed Sequential Data Sets

Table of Mutually Exclusive DD Parameters

	1 AFF	2 DCB	3 DDNAME	4 DEST	5 DISP	6 DLM	7 DSN	8 DSNAME	9 DUMMY	10 DYNAM	11 FCB	12 HOLD	13 LABEL	14 OUTLIM	15 QNAME	16 SEP	17 SPACE	18 SPLIT	19 SUBALLOC	20 SYSOUT	21 TERM**	22 TERM***	23 UCS	24 UNIT	25 VOL	26 VOLUME	27 *	28 DATA	
1 AFF	X																												
2 DCB		X																											
3 DDNAME	X		X																										
4 DEST			X	X																									
5 DISP			X	X	X																								
6 DLM						X																							
7 DSN			X				X																						
8 DSNAME			X				X	X																					
9 DUMMY									X																				
10 DYNAM	X		X						X	X																			
11 FCB			X								X																		
12 HOLD	X											X																	
13 LABEL			X									X																	
14 OUTLIM			X										X																
15 QNAME			X										X																
16 SEP	X														X														
17 SPACE			X													X													
18 SPLIT			X													X													
19 SUBALLOC			X													X													
20 SYSOUT	X		X									X									X								
21 TERM**			X																		X								
22 TERM***			X																		X								
23 UCS			X																			X							
24 UNIT			X																				X						
25 VOL			X																					X					
26 VOLUME			X																					X					
27 *	X		X																		X								
28 DATA			X																					X					

LEGEND:  The horizontal and the vertical DD parameters are mutually exclusive, i.e., they cannot be coded together on one DD statement.

** (For OS/VS1)
 *** (For OS/VS2)

As indicated by the table, each DD parameter is mutually exclusive with itself, i.e., it cannot appear twice on the same DD statement.

Figure 8. Table of Mutually Exclusive DD Parameters

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	
OLD or MOD or SHR	any except PASS	CATLG	requested disposition	deleted	cataloged	cataloged	
	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 cancelled 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 cancelled 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.

Figure 9. Disposition Processing Chart



Device	2314 (each volume)	2305	3330
Storage Medium	Disk	Disk	Disk
Cylinders	200	Model 1: 48 Model 2: 96	404
Tracks Per Cylinder	20	8	19
Bytes Per Track	7,294	Model 1: 14,136 Model 2: 14,660	13,030
Bytes Per Cylinder	145,880	Model 1: 113,088 Model 2: 117,280	247,570
Bytes Per Device (in millions)	29.17	Model 1: 5.4 Model 2: 11.3	101.6

Figure 10. 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		2314	2305-1	2305-2	3330
7294	14136	14660	13030	1	7249	13934	14569	12974
3520	6852	7231	6447	2	3476	6650	7140	6391
2298	4424	4754	4253	3	2254	4222	4663	4197
1693	3210	3516	3156	4	1649	3008	3425	3100
1332	2480	2773	2498	5	1288	2278	2682	2442
1092	1996	2278	2059	6	1049	1794	2187	2003
921	1648	1924	1745	7	877	1446	1833	1689
793	1388	1659	1510	8	750	1186	1568	1454
694	1186	1452	1327	9	650	984	1361	1271
615	1024	1287	1181	10	571	822	1196	1125
550	892	1152	1061	11	506	690	1061	1005
496	782	1040	962	12	452	580	949	906
450	688	944	877	13	407	486	853	821
411	608	863	805	14	368	406	772	749
377	538	792	742	15	333	336	701	686
347	478	730	687	16	304	276	639	631
321	424	676	639	17	277	222	585	583
298	376	627	596	18	254	174	536	540
276	334	584	557	19	233	132	493	501
258	296	544	523	20	215	94	453	467
241	260	509	491	21	198	58	418	435
226	230	477	463	22	183		386	407
211	200	448	437	23	168		357	381
199	174	421	413	24	156		330	357
187	150	396	391	25	144		305	335
176	128	373	371	26	133		282	315
166	106	352	352	27	123		261	296
157	88	332	335	28	114		241	279
148	70	314	318	29	105		223	262
139	52	297	303	30	96		206	247

Figure 11. Track Capacities

Indexes to OS/VS publications are consolidated in the *OS/VS Master Index*, GC28-0602, and the *OS/VS Master Index of Logic*, GY28-0603. For additional information about any subject listed below, refer to other publications listed for the same subject in the Master Index.

{ }
 use 13
 []
 use 13
 ...
 use 13
 &
 use 194
 &&
 use 194
 *parameter on DD statement 73-74
 coding BLKSIZE subparameter 73
 coding BUFNO subparameter 73
 examples of 74
 *in the PGM parameter 43
 *subparameter in the RESTART parameter 36
 *** 16
 *.ddname
 in the DCB parameter 83, 85
 in the DSNNAME parameter 140-142
 *.stepname.ddname
 in the DCB parameter 83, 85
 in the DSNNAME parameter 140-142
 in the PGM parameter 43-45
 *.stepname.proctepname.ddname
 in the DCB parameter 83, 85
 in the DSNNAME parameter 140-142
 in the PGM parameter 43-45
 /*
 use 175
 /**
 use 173
 ABEND dumps 69
 absolute track technique
 for ISAM data set 207
 ABSTR subparameter in the SPACE parameter 150-153
 for ISAM data set 207
 accessing messages received through TCAM 148
 accounting information
 (see accounting information parameter and ACCT parameter)
 accounting information parameter on JOB statement 23
 continuing 23
 example of 23
 format of 23
 rules for coding 23
 special characters in 23
 ACCT parameter on EXEC statement 47
 examples of 47
 format of 47
 overriding the 47
 rules for coding 47
 special characters in 47
 adding
 DD statements to cataloged procedures 206
 parameters to
 DD statements in cataloged procedures 62
 EXEC statements in cataloged procedures 41

address, unit 162
 address subparameter in the SPACE parameter 150-152
 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 25
 default 25
 examples of 25
 format of 25
 rules for coding 25
 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 150-153
 allocating space for data sets to share cylinders 154-155
 AL subparameter in the LABEL parameter 144.1, 190
 ALIGN subparameter of FCB parameter 143
 alphameric character set 17-18
 ALX subparameter in the SPACE parameter 150-153
 American National Standard labels 144.1-145
 ANSI printer control characters in RECFM subparameter 96, 104, 128
 ANSI tape labels 144.1, 145, 146
 apostrophes
 data set name in 19
 purpose 18-19
 appendixes 183-224
 area arrangement for ISAM data set 221
 ASCII magnetic tape
 (see DCB parameters)
 (see LABEL parameter)
 attributes, DCB (see DCB subparameters)
 AUL subparameter in the LABEL parameter 144.1-145
 automatic checkpoint restart (see restart facilities)
 automatic step restart (see restart facilities)
 automatic volume recognition (AVR)
 channel separation requests 149
 specifying a group name 162
 average block length
 in SPACE parameter 150
 in SPLIT parameter 154
 in SUBALLOC parameter 156
 AVR (see automatic volume recognition)
 backward reference 16-17
 to a concatenation 17
 in DCB parameter 16
 with deferred restart (see restart facilities)
 in DSNNAME parameter 16-17
 in PGM parameter 16
 in VOLUME parameter 16
 BFALN, DCB subparameter
 for BDAM 87

- for BISAM 91
- for BPAM 93
- for BSAM 97
- for EXCP 115
- for QISAM 117
- for QSAM 121
- BFTEK, DCB subparameter
 - for BTAM 107
 - for EXCP 109
 - for QSAM 121
- BISAM data set (*see indexed sequential data set*)
- blank
 - purpose 14-15
- BLKSIZE, DCB subparameter
 - coded with
 - * parameter 73-74
 - DATA parameter 75-76
 - DDNAME parameter 136-136.1
 - SPACE parameter 199
 - SUBALLOC parameter 204
 - for BDAM 87
 - for BPAM 93
 - for BSAM 97
 - for QISAM 117
 - for QSAM 122
 - for TCAM 131
- block length subparameter
 - in SPACE parameter 150-153
 - in SPLIT parameter 154-155
 - in SUBALLOC parameter 156-156.1
- 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 144.1
 - restriction on use 190
- braces
 - use 13
- brackets
 - use 13
- buffers
 - boundary (*see BFALN subparameter*)
 - length of (*see BUFL subparameter*)
 - number of (*see BUFNO subparameter*)
 - for all lines (*see BUFSIZE subparameter*)
 - for one line (*see BUFMAX subparameter*)
 - for receiving operation (*see BUFIN subparameter*)
 - for sending operation (*see BUFOUT subparameter*)
 - offset (*see BUFOFF subparameter*)
 - type (*see BFTEK subparameter*)
- BUFIN, DCB subparameter for TCAM 131
- BUFL, DCB subparameter
 - for BDAM 87
 - for BISAM 91
 - for BPAM 93
 - for BSAM 98
 - for EXCP 109
 - for QISAM 117
 - for QSAM 122
 - for TCAM 131
- BUFMAX, DCB subparameter for TCAM 132
- BUFNO, DCB subparameter
 - coded with
 - * parameter 73-74
 - DATA parameter 75-76
 - DDNAME parameter 136-136.1
 - for BDAM 88
 - for BISAM 91
 - for BPAM 94
 - for BSAM 98
 - for BTAM 107
 - for EXCP 109
 - for QISAM 118
 - for QSAM 122
- BUFOFF, DCB subparameter
 - for BSAM 98
 - for QSAM 123
- BUFOUT, DCB subparameter for TCAM 132
- BUFSIZE, DCB subparameter for TCAM 132
- 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 171
- cataloged data set
 - generation data set 210-214
 - label type information 212, 213
 - unit information 211, 213
 - retrieving 212-213
- cataloged procedure
 - adding to procedure library 196
 - assigning values to symbolic parameters 194-195
 - calling 46
 - contents of 193
 - DD statement 193-196
 - adding DD statements 196
 - overriding parameters on 41, 196
 - EXEC statement 41-42
 - adding parameters to 41
 - overriding parameters on 41
 - modifying 193-196
 - using 193-196
 - writing 193-196
- CATLG subparameter in the DISP parameter 137
- channel affinity (*see AFF parameter*)
- channel separation 80
- character set
 - alphanumeric 17-18
 - national 17-18
 - special 17-19
- character set for output data set 160.1
- checkid subparameter in the RESTART parameter 36
 - special characters in 36
- checkpoint data set 71
- checkpoint/restart facilities
 - automatic 30-31
 - checkid 36
 - checkpoint data set 71-72
 - checkpoint restart 33-34, 36-37, 56-57, 71-72
 - deferred checkpoint restart 33-34, 56-57
 - deferred step restart 33-34, 56-57
 - RD parameter on EXEC statement 56-57
 - RD parameter on JOB statement 33-34
 - RESTART parameter on JOB statement 36-37
 - SYSCHK DD statement 71-72
- CHKPT macro instruction 33-34, 56-57
- class
 - job 26
 - message 28
 - system output 157-158
- CLASS parameter on JOB statement 26
 - assigning a job class 26
 - default 26
 - examples of 26
 - format of 26
 - rules for coding 26
- classnames
 - for output streams 157
- CODE, DCB subparameter
 - for BSAM 98
 - for EXCP 110
 - for QSAM 123
- coding special characters 18-19
- comma
 - purpose 13

command statement 171-172
 commands 171
 example of 172
 format of 171
 rules for coding 172
 commands, operator 171
 comment statement 173
 example of 173
 format of 173
 rules for coding 173
 comments field 14, 16
 continuation of 16
 completing information in a data control block 83-85
 completion codes
 use of 27, 49-50
 concatenating data sets 17
 concatenation
 of data sets 17
 of private libraries 63-68
 COND parameter on EXEC statement 49-50
 examples of 50
 format of 49
 overriding 50
 rules for coding 49-50
 use of 49
 COND parameter on JOB statement 27
 examples of 27
 format of 27
 rules for coding 27
 use of 27
 conditional disposition of a data set 137-139
 CATLG 137, 138
 DELETE 137
 KEEP 137, 138
 UNCATLG 137, 138
 conditional execution
 of a job 27
 of a job step 49-50
 CONTIG subparameter in the SPACE parameter 150-152
 continuing control statements 15-16
 comments field 15-16
 operand field 15-16
 control volume 185
 COPIES parameter 82
 CPU time limit
 on EXEC statement 59
 on JOB statement 38
 creating data sets 185-192
 nontemporary 185-186
 temporary 186-187
 CYL subparameter
 in SPACE parameter 150
 in SPLIT parameter 154
 in SUBALLOC parameter 156
 cylinders
 sharing 201, 202
 CYLOFL, DCB subparameter for QISAM 118

data control block
 completing the 83-85
 data definition statement
 (see DD statement)
 data in the input stream
 defining 73-76
 DATA parameter on DD statement 75-76
 coding BLKSIZE subparameter 75
 coding BUFNO subparameter 75
 examples of 76
 format of 75
 rules for coding 75
 data set
 creating (see creating data sets)
 retrieving (see retrieving data sets)

data set in the input stream
 defining 73-76
 data set label
 completing the data control block 83-85
 copying attributes from a 84-85
 model 210-211
 data set name 141-142
 in apostrophes 141
 copying name from earlier DD statement 141-142
 nontemporary 141
 qualified 141-142
 temporary 141
 unqualified 141-142
 data set organization (see DSORG subparameter)
 DATETIME macro (see RESERVE subparameter)
 DCB
 (see data control block)
 DCB attributes
 (see DCB subparameters)
 DCB macro instruction
 completing the data control block 83-85
 DCB parameter on DD statement 83-85
 backward references to 85
 coded on
 JOBLIB DD statement 64
 STEPLIB DD statement 67
 SYSCHK DD statement 71-72
 coded when
 creating generation data set 212
 creating ISAM data set 206
 retrieving generation data set 213
 retrieving ISAM data set 209
 coded with
 * parameter 73
 DATA parameter 75
 DDNAME parameter 135
 completing the data control block 83-85
 copying information from
 data set label 84
 earlier DD statement 85
 examples of 85
 format of 83
 overriding subparameters in 84-85
 rules for coding 83-85
 subparameters
 for BDAM 87-90
 for BISAM 91-92
 for BPAM 93-96
 for BSAM 97-106
 for BTAM 107-108
 for EXCP 109-114
 for GAM 115
 for QISAM 117-120
 for QSAM 121-130
 for TCAM 131-134
 DD statement 61-62
 examples of 62
 format of 61
 keyword parameters 61
 overriding parameters on 62
 positional parameters 61
 rules for coding 61
 ddname
 assigning 61
 examples of 62
 qualified (see *.ddname)
 special 61
 DDNAME parameter on DD statement 136-136.1
 coded with
 BLKSIZE subparameter 135
 BUFNO subparameter 135
 examples of 136-136.1
 format of 136
 rules for coding 136

- dedicated data set
 - using 188
- default for
 - CLASS parameter 26
 - CPU time limit 38, 59
 - DPRTY parameter 52
 - MSGCLASS parameter 28
 - MSGLEVEL parameter 29
 - output class for system messages 28
 - PRTY parameter 32
 - REGION parameter 35, 58
 - region size 35, 58
 - UNIT parameter 163
 - TIME parameter 38, 59
 - wait-state time limit 38, 59
- DEFER subparameter in the UNIT parameter 162, 163
- 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 a private library for a job 63-65
- defining a private library for a job step 66-68
- defining restart
 - on EXEC statement (*see* RD parameter)
 - on JOB statement (*see* RD parameter)
- DELETE subparameter in the DISP parameter 137
- delimiter statement 175
 - * parameter 175
 - DATA parameter 175
 - example of 175
 - format of 175
 - rules for coding 175
- delimiter other than /* 140
- DEN, DCB subparameter
 - for BSAM 99
 - for EXCP 110-111
 - for QSAM 124
- devices
 - capacities 224
- DIAGNS, DCB subparameter
 - for BDAM 88
 - for BISAM 92
 - for BPAM 94
 - for BSAM 100
 - for BTAM 107
 - for EXCP 111
 - for GAM 115
 - for QISAM 118
 - for QSAM 124
- directory
 - requesting space for
 - in SPACE parameter 199-200
 - in SUBALLOC parameter 204
- DISP parameter on DD statement 137-139
 - coded on
 - JOBLIB DD statement 63-65
 - STEPLIB DD statement 66-68
 - SYSABEND DD statement 69-70
 - SYSCHK DD statement 71-72
 - SYSUDUMP DD statement 69-70
 - coded when
 - creating generation data set 211
 - creating ISAM data set 206-207
 - retrieving generation data set 211
 - retrieving ISAM data set 209
 - conditional disposition subparameter 137-138
 - disposition subparameter 137-138
 - examples of 138-139
 - format of 137
 - rules for coding 138
- disposition of a data set 137-139
 - CATLG 137, 138
 - conditional disposition 137-138
 - DELETE 137
 - KEEP 137, 138
 - PASS 137
 - UNCATLG 137, 138
- disposition processing chart 223
- DLM parameter 140
 - examples of 140
 - format of 140
 - rules for coding 140
- DPRTY parameter on EXEC statement 52-53
 - examples of 53
 - format of 52
 - rules for coding 52
- DSN parameter on DD statement (*see* DSNAME parameter)
- DSNAME parameter on DD statement 141-142
 - backward references 141-142
 - coded on
 - JOBLIB DD statement 63-65
 - STEPLIB DD statement 66-68
 - SYSABEND DD statement 69-70
 - SYSCHK DD statement 71-72
 - SYSUDUMP DD statement 69-70
 - coded when
 - creating generation data set 211
 - creating ISAM data set 205
 - retrieving generation data set 212
 - retrieving ISAM data set 208
 - copying name from earlier DD statement 188-189
 - examples of 142
 - format of 141
 - name in apostrophes 142
 - nontemporary data set names 185-186
 - rules for coding 142
 - special characters in 142
 - temporary data set names 186-187
- DSORG, DCB subparameter
 - for BDAM 88
 - for BISAM 92
 - for BPAM 94
 - for BSAM 100
 - for BTAM 108
 - for EXCP 111
 - for GAM 115
 - for QISAM 118
 - for QSAM 124
- dummy data set 77-78
- DUMMY parameter on DD statement 77-78
 - examples of 78
 - format of 77
 - nullifying 77
 - rules for coding 77
- dump, abnormal termination 69
 - storing 69
 - writing to unit record 69
- DYNAM parameter on DD statement 79
 - examples of 79
 - format of 79
 - rules for coding 79
- ellipsis
 - use 14
- EROPT, DCB subparameter
 - for BTAM 108
 - for QSAM 125
- error option (*see* EROPT subparameter)
- EVEN subparameter in the COND parameter 27, 49-50
- EXEC statement 41-42
 - examples of 42
 - fields in 41
 - format of 41

- keyword parameters on 41
- positional parameters on 41
- rules for coding 41
- execute statement (*see* EXEC statement)
- execution
 - of a cataloged procedure 46
 - of a processing program 43-45
- EXPDT subparameter in the LABEL parameter 145
- expiration date 144.1-145
(*see also* retention period)
- extending a data set 218, 220

- FCB parameter 143
 - examples of 143
 - format of 143
 - image identifier 143
 - requesting alignment of forms 143
 - rules for coding 143
- fields 14-16
 - comments 16
 - examples of 14
 - name 14
 - operand 14, 15
 - operation 14
- fixed-length record (*see* RECFM subparameter)
- FOLD subparameter in the UCS parameter 160.1-161
- form number subparameter in the SYSOUT parameter 157-158
- format of
 - command statement 171
 - comment statement 173
 - DD statement 61
 - delimiter statement 175
 - EXEC statement 41
 - JOB statement 21
 - null statement 177
 - PEND statement 179
 - PROC statement 181
 - publication 3
- forms control image 143
- FUNC, DCB subparameter
 - for BSAM 100-101
 - for QSAM 125

- generation data group (*see* generation data set)
- generation data set 210-214
 - creating 210-212
 - with deferred restart 213
 - name of 211
 - retrieving 212-213
- generation numbers, relative 186, 212
- GNCB, DCB subparameter for GAM 115
- group name subparameter in the UNIT parameter 162

- HIARCHY subparameter in the DCB parameter 3
- HOLD parameter on DD statement 144
- HOLD subparameter in the TYPRUN parameter 40
- holding a job for execution 40

- identifying the data set 183-192
- IEFBR14 program 45
- IN subparameter in the LABEL parameter 145
- incremental quantity (*see* secondary quantity)
- index
 - requesting space for 199-200
- index area 205-208, 221
- indexed sequential data set
 - area arrangement of 208, 221
 - creating 205-208
 - example of 209

- lengthening 220
- name
 - nontemporary 205-206
 - temporary 205-206
- requesting space for index 199-200
- retrieving 208-209
 - example of 209
- unit restrictions 206, 208
- input/output macro instructions (*see* GNCB subparameter)
- input data set
 - concatenating 16
 - identifying the data set 183-192
 - IN subparameter 145
 - 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 73, 76
- installation requirements 23
- in-stream procedures
 - assigning values to symbolic parameters 194-196
 - calling 46
 - contents of 193
 - DD statement 193
 - adding DD statements 193
 - adding parameters to 193
 - nullifying parameters 193
 - overriding parameters on 193
 - EXEC statement 193
 - adding parameters to 193
 - nullifying parameters 193
 - overriding parameters to 193
 - modifying 196
 - using 193
 - writing 194-196
- ISAM data set (*see* indexed sequential data set)

- job priority 32
- job class 26
 - default 26
 - priority 26
- job library 63-65
- JOB statement 21-22
 - examples of 22
 - fields in 21
 - format of 21
 - keyword parameters on 21
 - positional parameters on 21
- job step 41
- jobclass subparameter in the CLASS parameter 26
- JOBLIB DD statement 63-65
(*see also* STEPLIB)
 - examples of 64-65
 - parameters to code when
 - cataloged 63-64
 - not cataloged 64
 - rules for coding 63-64
- jobname 21
 - assigning 21
 - examples of 22

- KEEP subparameter in the DISP parameter 137-138
- KEYLEN, DCB subparameter
 - for BDAM 89
 - for BPAM 94
 - for BSAM 101
 - for EXCP 112
 - for QISAM 119

keylength (*see* KEYLEN subparameter)

keyword parameters

- on DD statement 61
- on EXEC statement 41
- on JOB statement 21
- rules for coding 15

LABEL parameter on DD statement 144.1-146

- coded on SYSCHK DD statement 71
- coded when
 - creating generation data set 212
 - creating ISAM data set 206
 - retrieving generation data set 213
- data set sequence number subparameter 144.1
- examples of 146
- EXPDT subparameter 145
- format of 144.1
- IN subparameter 145
- label type subparameter 144.1
- OUT subparameter 145
- PASSWORD subparameter 145
- RETPD subparameter 145
- rules for coding 145-146

label types 144.1-146

labels

- data set 144.1
- direct access 144.1
- nonstandard (NSL) 144.1
- standard (SL) 144.1
- standard and user (SUL) 144.1
- tape 144.1

lengthening a data set 218, 219, 220

libraries, concatenating private (*see* JOBLIB and STEPLIB)

library

- private 45
 - for a job 63-65
 - for a step 66-68
- system 44
- temporary 44

LIMCT, DCB subparameter for BDAM 89

limit for output records 147

LOG command 171

logical record length (*see* LRECL subparameter)

LRECL, DCB subparameter

- for BPAM 95
- for BSAM 101
- for QISAM 119
- for QSAM 126
- for TCAM 132

LTM subparameter in the LABEL parameter 144.1

message queue records (*see* THRESH subparameter)

MOD subparameter in the DISP parameter 137

MODE, DCB subparameter

- for BSAM 102
- for EXCP 112
- for QSAM 126

mode for card reader/punch (*see* MODE subparameter)

MODIFY command 171

MOUNT command 171

mounting

- deferred 162, 163
- parallel 162, 163

MSGCLASS parameter on JOB statement 28

- assigning an output class 28
- coded with SYSOUT parameter 28
- default 28
- examples of 28
- format of 28
- rules for coding 28

MSGLEVEL parameter on JOB statement 29

- default 29
- examples of 29
- format of 29
- rules for coding 29

MXIG subparameter in the SPACE parameter 151-152

name field 14

- example of 14

national character set 17-18

NC subparameter in the RD parameter

- on EXEC statement 56
- on JOB statement 33

NCP, DCB subparameter

- for BISAM 92
- for BPAM 95
- for BSAM 102

NEW subparameter in the DISP parameter 137

NL subparameter in the LABEL parameter 144.1

nonspecific volume request 167, 169

- for direct access volume 169
- satisfying a 169
- for tape volume 169

nonstandard labels 144.1

nontemporary data set

- creating 185-186

NOTIFY parameter on JOB statement 31

- example of 31
- format of 31
- rules for coding 31

NOPWREAD subparameter in the LABEL parameter 145

NR subparameter in the RD parameter

- on EXEC statement 56
- on JOB statement 33

NSL subparameter in the LABEL parameter 144.1

NTM, DCB subparameter for QISAM 119

null statement 177

- example of 177
- format of 177

NULLFILE (*see* DUMMY parameter)

OLD subparameter in the DISP parameter 137

ONLY subparameter in the COND parameter

- on EXEC statement 49-50
- on JOB statement 27

OPEN/CLOSE/EOV trace option (*see* DIAGNS subparameter)

operand field 14-15

- blank 14
- example of 14
- keyword parameters 15
- positional parameters 15
- subparameters 15

operation field 14-15

operator commands 171-172

operator subparameter in the COND parameter

- on EXEC statement 49-50
- on JOB statement 27

OPTCD, DCB subparameter

- for BDAM 89
- for BPAM 95
- for BSAM 103
- for EXCP 112
- for QISAM 119-120
- for QSAM 127
- for TCAM 133

optional services (*see* OPTCD subparameter)

OUT subparameter in the LABEL parameter 145

OUTLIM parameter 147

- coded with SYSOUT parameter 147

- determining the output limit 147
- example 147
- rules for coding 147
- output of
 - allocation messages 29
 - allocation recovery messages 29
 - disposition messages 29
 - job control statements 29
- output class
 - for output data set 157
 - for system messages 28
- output class subparameter in the MSGCLASS parameter 28
- output data set
 - allocating space for 150-156.1
 - creating 216
 - lengthening 218
 - OUT subparameter 145
 - printed using UCS feature (*see* UCS parameter)
 - routed through output stream (*see* SYSOUT parameter)
 - specifying
 - conditional disposition (*see* COND parameter)
 - disposition (*see* DISP parameter)
 - status (*see* DISP parameter)
 - unit information (*see* UNIT parameter)
 - volume information (*see* VOLUME parameter)
- output stream
 - routing data sets through the 157-158
- output writer 157-158
- overflow area 205-208, 221

- P subparameter in the UNIT parameter 162-163
- parentheses
 - to enclose a subparameter list 15
 - inclusion in variables 15
- PARM parameter on EXEC statement 54-55
 - examples of 55
 - format of 54
 - overriding the 54
 - rules for coding 54
 - special characters in 54
- partition 35, 58
- partitioned data set
 - concatenating 16
 - creating 216
 - executing programs in 46
 - lengthening 218
 - name
 - nontemporary 185-186
 - temporary 186-188
 - retrieving a member of 186
 - space for directory
 - in SPACE parameter 199-200
 - in SUBALLOC parameter 204
- PASS subparameter in the DISP parameter 137
- passing variable information to a program 54-55
- password protection 144.1
- PASSWORD subparameter in the LABEL parameter 145, 146
- PCI, DCB subparameter for TCAM 133
- PEND statement 179
- PGM parameter on EXEC statement 43-45
 - backward references 43
 - examples of 43-44
 - executing programs from
 - private library 45
 - system library 44
 - temporary library 44
 - format of 43
- positional parameters
 - on DD statement 61
 - on EXEC statement 41
 - on JOB statement 21
 - rules for coding 15

- postponing definition of a data set (*see* DDNAME parameter)
- primary quantity
 - in SPACE parameter 150-153, 198
 - in SPLIT parameter 154-155
 - in SUBALLOC parameter 155-156.1, 203
- prime area 205-209
- priority
 - job 32
 - job class 26
 - job step 52-53
- priority parameter (*see* PRTY parameter)
- private libraries 45, 63, 66
 - concatenating (*see* JOBLIB and STEPLIB)
 - defining 63, 66
 - executing programs from 43-45
- PRIVATE subparameter in the VOLUME parameter 165
- PROC parameter on EXEC statement 46
 - examples of 46
 - format of 46
 - rules for coding 46
- PROC statement 181-182
 - assigning values to symbolic parameters on 194-196
 - example of 182
 - format of 181
 - rules for coding 181-182
- procedure
 - (*see* cataloged procedure and instream procedure)
- program
 - calling 43-45
- program controlled interruption (*see* PCI subparameter)
- program name
 - subparameter in the SYSOUT parameter 157
- programmer's name parameter on JOB parameter 24
 - examples of 24
 - format of 24
 - rules for coding 24
 - special characters in 24
- PRTSP, DCB subparameter
 - for BSAM 104
 - for EXCP 113
 - for QSAM 128
- PRTY parameter on JOB statement 32
 - default 32
 - examples of 32
 - format of 32
 - rules for coding 32

- QISAM data set
 - DCB subparameters 117-120
 - (*see also* ISAM data set)
- QNAME parameter on the DD statement 148
 - example of 148
 - format of 148
 - rules for coding 148
- qualified name
 - assigning 185
- R subparameter in the RD parameter
 - on EXEC statement 56
 - on JOB statement 33
- RD parameter on EXEC statement 56-57
 - defining restart 56
 - examples of 57
 - format of 56
 - overriding the 57
 - restart facilities 56
 - rules for coding 57
- RD parameter on JOB statement 33-34
 - defining restart 33
 - examples of 34
 - format of 33

- overriding the 34
- restart facilities 33
- rules for coding 34
- READ/WRITE macros before a CHECK macro (*see* NCP subparameter)
- REAL subparameter in the ADDRSPC parameter
 - on the EXEC statement 48
 - on the JOB statement 25
- real address space 25, 48
- real storage 25, 48
- RECFM, DCB subparameter
 - for BDAM 90
 - for BPAM 96
 - for BSAM 104
 - for QISAM 120
 - for QSAM 128
 - for TCAM 134
- 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 165, 167
- references, backward 16
- region size 35, 58
- REGION parameter on EXEC statement 58
 - default 58
 - examples of 58
 - format of 58
 - overriding the 58
 - rules for coding 58
- REGION parameter on JOB statement 35
 - default 35
 - examples of 35
 - format of 35
 - rules for coding 35
- relational operators in the COND parameter
 - on the EXEC statement 49-50
 - on the JOB statement 27
- relative generation number 210
- relative track number 197
- RELEASE command 171
- releasing unused space (*see* RLSE)
- removable volume 168-169
- REPLY command 171
- REPOS, DCB subparameter for EXCP 113
- repositioning for tape devices (*see* REPOS subparameter)
- requesting channel separation 149
- requesting copies of an output data set 82
- RESET command 171
- RESERVE, DCB subparameter for TCAM 134
- reserve volume 168-169
- restart definition (RD parameter)
 - on EXEC statement 56-57
 - on JOB statement 33-34
- restart facilities
 - RD parameter on EXEC statement 56-57
 - RD parameter on JOB statement 33-34
 - RESTART parameter on JOB statement 36-37
 - REPOS, DCB subparameter 113
- RESTART parameter on JOB statement 36-37
 - examples of 37
 - format 36
 - rules for coding 36-37
 - when defining generation data set 37
 - when making backward reference 37
- RETAIN subparameter in the VOLUME parameter 165, 166
- retention period 145, 146
- RETPD subparameter in the LABEL parameter 145, 146
- retrieving data sets 217, 220
 - generation data set 212-214
 - indexed sequential data set 208-209
 - member of partitioned data set 185, 186
- return code test 27, 49-50.
- RKP, DCB subparameter for QISAM 120
- RLSE subparameter in the SPACE parameter 150
- RNC subparameter in the RD parameter
 - on EXEC statement 56
 - on JOB statement 33
- ROLL parameter
 - on EXEC statement 3
 - on JOB statement 3
- ROUND subparameter in the SPACE parameter 151-152
- SCAN, subparameter in the TYPRUN parameter 40
- scanning JCL for syntax errors 40
- scratch volume 168-169
- secondary quantity
 - in SPACE parameter 150, 198-199
 - in SPLIT parameter 154, 201
 - in SUBALLOC parameter 156, 204
- SEP parameter in DD statement 149
 - examples of 149
 - format of 149
 - requesting channel separation 149
 - rules for coding 149
- SEP subparameter in the UNIT parameter 162, 163
- separation
 - channel 80, 149
 - unit 162, 163
- SEQUENCE macro (*see* RESERVE subparameter)
- sequence number
 - data set 144
 - volume 165, 166
- sequential data set
 - concatenating 17
 - creating 216
 - lengthening 218
 - retrieving 217
- SER subparameter in the VOLUME parameter 165, 167
- SET command 171
- sharing
 - cylinders (*see* SPLIT parameter)
 - data set (*see* DISP parameter)
- shortening processing time 149
- SHR subparameter in the DISP parameter 137
- SL subparameter in the LABEL parameter 144.1
- SPACE parameter on DD statement 150-153, 197-201 (*see also* SPLIT and SUBALLOC)
 - assigning specific tracks 197-198
 - coded on
 - SYSABEND DD statement 69
 - SYSUDUMP DD statement 69
 - coded when
 - creating generation data set 212
 - creating ISAM data set 207
 - coded with SYSOUT parameter 157
 - examples of 153
 - format of 150
 - letting system assign specific tracks 197-198
 - allocating whole cylinders 201
 - releasing unused space 200
 - requesting space for directory 199
 - requesting space for index 199
 - specifying format 200
 - specifying primary quantity 198
 - specifying secondary quantity 198-199
 - unit of measurement 198
 - rules for coding 151
- space on a printer (*see* PRTSP subparameter)
- special character set
 - with UCS parameter 160.1
 - using 17-18
- special ddnames 61
- specific volume request 168

split cylinder mode 201

SPLIT parameter on DD statement 154-155
(see also SPACE and SUBALLOC)
 coded on
 SYSABEND DD statement 69
 SYSUDUMP DD statement 69
 examples of 155
 format of 154
 requesting space 201-202
 rules for coding 154-155

STACK, DCB subparameter
 for BSAM 105
 for EXCP 113
 for QSAM 129

stacker bin *(see STACK subparameter)*

START command 171

step restart
 automatic 33, 56
 deferred 36-37

STEPLIB DD statement 66-68
(see also JOBLIB)
 concatenating private libraries 66
 examples of 67-68
 parameters to code when
 cataloged 67-68
 not cataloged or not passed 68
 passed 68
 rules for coding 67-68

stepname
 assigning 41
 examples of 42

STOP command 171

storage volume 169

SUBALLOC parameter on DD statement 156-156.1
(see also SPACE and SPLIT)
 coded on
 SYSABEND DD statement 69
 SYSUDUMP DD statement 69
 examples of 156.1
 format of 156
 requesting space 202-204
 rules for coding 156

suballocation *(see SUBALLOC parameter)*

SUL subparameter in the LABEL parameter 144.1

suppressing
 CHKPT macro instruction 33-34, 56-57
 automatic restarts 33-34, 56-57

symbolic parameters
 assigning default values to 194
 assigning values to 194-195
 defining 194-196
 definition of 194
 examples of 194
 nullifying 196
 PROC statement 195-196

SYSABEND DD statement 69-70
(see also SYSUDUMP)
 examples of 69-70
 storing the dump 69
 writing the dump to unit record device 69

SYSCHK DD statement 71-72
 examples of 72
 parameters to code when
 cataloged 71
 not cataloged 72
 rules for coding 71-72

SYSOUT parameter on DD statement 157-158
 coded on
 SYSABEND DD statement 69
 SYSUDUMP DD statement 69
 examples of 157-158
 format of 157

rules for coding 157
 specifying classname 157
 specifying DCB parameter 157
 specifying form number 157
 specifying program name 157
 specifying SPACE parameter 157
 specifying UNIT parameter 157

system library 44

System Management Facilities
 with TIME parameter 38-39, 59-60

system messages
 output class 28

SYSUDUMP DD statement 69-70
(see also SYSABEND)
 examples of 69-70
 storing the dump 69
 writing the dump to unit record device 69

SYS1.LINKLIB 44

tape density *(see DEN subparameter)*

tape labels, ANSI 144.1-146

TCAM *(see Telecommunications Access Method)*

Telecommunications Access Method (TCAM)
 DCB subparameters 131-134

temporary data set, creating 186-188

TERM parameter (for OS/VS1) on DD statements 159

temporary library 44

TERM parameter (for OS/VS2) on DD statements 160
 examples of 160
 format of 160
 rules for coding 160

THRESH, DCB subparameter for TCAM 134

TIME parameter on EXEC statement 59-60
 CPU time limit 59
 default 59
 effect of JOB limit 59
 eliminating timing 60
 examples of 60
 format of 59
 overriding the 59
 rules for coding 59
 wait-state time limit 59, 60
 1440 59

TIME parameter on JOB statement 38-39
 CPU time limit 38
 default 38
 effect of JOB limit 38
 eliminating timing 38
 examples of 39
 format of 38
 rules for coding 38
 wait-state time limit 38
 1440 38

timing
 CPU 38, 59
 eliminating 38, 60

track number, relative 197

tracks for cylinder index *(see NTM subparameter)*

tracks for overflow *(see CYLOFL subparameter)*

tracks to be searched *(see LIMCT subparameter)*

TRK subparameter
 in SPACE parameter 150
 in SUBALLOC parameter 156

TRTCH, DCB subparameter
 for BSAM 105-106
 for EXCP 113-114
 for QSAM 129-130

TYPRUN parameter on JOB statement 40
 example of 40
 format of 40
 rules for coding 40



UCS parameter on DD statement 160.1-161
 examples of 161
 format of 160.1
 identifying character set 160.1
 requesting
 fold mode 160.1
 operator verification 160.1
 rules for coding 160.1
 special character sets 160.1
 UNCATLG subparameter in the DISP parameter 137, 138
 unit address 162, 163
 unit affinity 162
 unit count subparameter in the UNIT parameter 162-163
 UNIT parameter on DD statement 162-164
 coded on
 JOB LIB DD statement 64
 STEPLIB DD statement 67
 SYSABEND DD statement 69
 SYSCHK DD statement 71-72
 SYSUDUMP DD statement 69
 coded when
 creating generation data set 211
 creating ISAM data set 206
 retrieving generation data set 213
 retrieving ISAM data set 208
 examples of 164
 format of 162
 identifying the device 162-163
 providing unit information 162-163
 rules for coding 163-165
 specifying
 deferred mounting 162-163
 parallel mounting 162-163
 unit affinity 162-163
 unit count 162-163
 unit separation 163-164
 use with TERM parameter 163
 unit record devices, writing dumps to 69-70
 unit separation 162-164
 universal character set (*see* UCS)
 UNLOAD command 171

V format (*see* RECFM subparameter)
 VARY command 171
 VERIFY subparameter
 of FCB parameter 143
 of UCS parameter 160.1

VIRT subparameter in the ADDRSPC parameter
 on the EXEC statement 48
 on the JOB statement 25
 virtual address space 25, 48
 virtual storage 25, 48
 VOL parameter on DD statement (*see* VOLUME parameter)
 volume (*see* VOLUME parameter)
 volume count subparameter in the VOLUME parameter 165, 166
 VOLUME parameter on DD statement 165-169
 backward reference 165-167
 coded on
 JOB LIB DD statement 64
 STEPLIB DD statement 67
 SYSABEND DD statement 69
 SYSCHK DD statement 71-72
 SYSUDUMP DD statement 69
 coded when
 creating generation data set 211
 creating ISAM data set 206
 retrieving ISAM data set 209
 examples of 168
 format of 165
 providing volume information 165-169
 referring to specific request 168
 rules for coding 166-167
 specifying
 PRIVATE subparameter 165, 166, 168, 169
 RETAIN subparameter 165, 166
 volume sequence number subparameter 165, 166
 volume count subparameter 165, 166
 with suballocation (*see* SUBALLOC parameter)
 supplying serial numbers 165, 167
 volume sequence number subparameter in the VOLUME parameter 165, 166
 volume serial number
 special characters in 165, 167
 VOLUME=REF
 backward references 165-167

wait state time limit 38, 60
 writing output messages 28, 29

XX* 16

1440 38, 59

The DD Statement				
//Name	Operation	Operand	P/K	Comments
// [ddname, procstepname, ddname]	DD	* [DATA [DLM=xx]]	P	Defines data set in the input stream
		[DUMMY]	P	Bypasses I/O operations on a data set (BSAM and QSAM)
		[DYNAM] 2	P	
		[AFF=ddname]	K	Requests channel separation
		[COPIES=nnn]	K	For use with the SYSOUT parameter
		DCB=(list of attributes)	K	Completes the data control block
		DCB={ dsname, *ddname, *stepname, ddname, *stepname, procstepname, ddname } [list of attributes]	K	
		[DDNAME=ddname]	K	Postpones the definition of a data set
		[DEST=userid]'	K	Specifies remote destination for output data set
		DISP={ NEW, OLD, SHR, MOD } { DELETE, KEEP, PASS, UNCATLG } { DELETE, KEEP, CATLG, UNCATLG }	K	Assigns a status, disposition, and conditional disposition to the data set
		[DLM=delimiter]	K	Assigns delimiter other than */
		{ DSNAME, DSN } = { dsname(member name), dsname(generation number), dsname(area name), &dsname, &dsname(member name), &dsname(area name), *ddname, *stepname, ddname, *stepname, procstepname, ddname }	K	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.
		[FCB=(image-id, ALIGN, VERIFY)]	K	Specifies forms control information. The FCB parameter is ignored if the data set is not written to a 3211 printer.
[HOLD={YES, NO}] 1	K	Specifies whether output processing is to be deferred or processed normally		
LABEL=([data set seq #] [SL, SUL, AL, AUL, NSL, NL, BLP, LTM], [PASSWORD, NOPWREAD], [IN, OUT], [EXPDT=yyddd], [RETPD=nnnn])	K	Supplies label information		

Chart 3

The DD Statement (con't)				
//Name	Operation	Operand	P/K	Comments
// [ddname, procstepname, ddname]	DD	[OUTLIM=number] 1	K	Limits the number of logical records you want included in the output data set
		[QNAME=process name]	K	Specifies the name of a TPROCESS macro which defines a destination queue for messages received by means of TCAM
		[SEP=(ddname, ...)]'	K	Requests channel separation
		{ SPACE={ TRK, CYL } (primary, secondary) [directory] [RLSE] [CONTIG] [ALX] [ROUND] }	K	Assigns space on a direct access volume for a new data set
		{ SPACE={ ABSTR, (primary quantity, address [directory]) }	K	Assigns specific tracks on a direct access volume for a new data set
		SPLIT={ (n, CYL, (primary quantity [secondary quantity])), (percent, blocklength, (primary quantity [secondary quantity])) }	K	Assigns space on a direct access volume for a new data set. Data sets share cylinders
		SUBALLOC={ TRK, CYL } (primary, secondary) [directory] { ddname, stepname, ddname, stepname, procstepname, ddname }	K	Requests part of the space on a direct access volume assigned earlier in the job
		[SYSOUT=(class name, [program name], [form number]), [OUTLIM=number]]	K	Routes a data set through the output stream. For class name, assign A-Z or 0-9
		[TERM=RT] 1	K	Indicates that an RTAM device is in use
		[TERM=TS] 2	K	Identifies TSO user
		[UCS=(character set code [FOLD] [VERIFY])]	K	Requests a special character set for a 1403 printer
		{ UNIT={ [unit address, unit count] [device type, P] [group name], [DEFER] [SEP=(ddname, ...)] }	K	Provides the system with unit information
		{ VOLUME={ [PRIVATE] [RETAIN] [volume seq #] [volume count] [SER=(serial number, ...), REF=dsname, REF=*ddname, REF=*stepname, ddname, REF=*stepname, procstepname, ddname] }	K	Provides the system with volume information

Legend:
P Positional parameter.
K Keyword parameter.
{} Choose one.
[] Enclosing subparameter, indicates that subparameter is optional; if more than one line is enclosed, choose one or more.
[] Enclosing entire parameter, indicates that parameter may be optional, depending on what type of data set you are defining.
1 For OS/V51 only.
2 For OS/V52 only.

Figure 14. The DD Statement

OS/VS JCL Reference

GC28-0618-2

**READER'S
COMMENT
FORM**

Your views about this publication may help improve its usefulness; this form will be sent to the author's department for appropriate action. Using this form to request system assistance or additional publications will delay response, however. For more direct handling of such requests, please contact your IBM representative or the IBM Branch Office serving your locality.

Possible topics for comment are:

Clarity Accuracy Completeness Organization Index Figures Examples Legibility

Cut or Fold Along Line

What is your occupation? _____

Number of latest Technical Newsletter (if any) concerning this publication: _____

Please indicate in the space below if you wish a reply.

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. Elsewhere, an IBM office or representative will be happy to forward your comments.

Cut or Fold Along Line

Your comments, please . . .

This manual is part of a library that serves as a reference source for system analysts, programmers, and operators of IBM systems. Your comments on the other side of this form will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

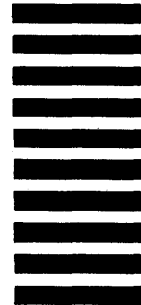
Fold

Fold

**First Class
Permit 170
Endicott
New York**

Business Reply Mail

No postage stamp necessary if mailed in the U.S.A.



Postage will be paid by:

**International Business Machines Corporation
Department G60
P. O. Box 6
Endicott, New York 13760**

Fold

Fold



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

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



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

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