IBM

MVS/Extended Architecture Data Administration: Macro Instruction Reference

Licensed Program

Order Number
GC26-4141-2

Data Facility Product
5665-XA2

Version 2
Release 3.0

IBM

MVS/Extended Architecture Data Administration: Macro Instruction Reference

Licensed Program

Order Number
GC26-4141-2

Data Facility Product
5665-XA2

Version 2
Release 3.0

Third Edition (June 1987)

This is a major revision of, and makes obsolete, GC26-4141-1.

This edition applies to Version 2 Release 3.0 of MVS/Extended Architecture Data Facility Product, Licensed Program 5665-XA2, and to any subsequent releases until otherwise indicated in new editions or technical newsletters.

The changes for this edition are summarized under "Summary of Changes" following the preface. Specific changes are indicated by a vertical bar to the left of the change. These bars will be deleted at any subsequent republication of the page affected. Editorial changes that have no technical significance are not noted.

Changes are made periodically to this publication; before using this publication in connection with the operation of IBM systems, consult the latest _IBM System/370, 30xx, and 4300 Processors Bibliography_, GC20-0001, for the editions that are applicable and current.

PREFACE

This publication contains information required by system and application programmers to use the data management access method functions of MVS/Extended Architecture Data Facility Product, Program Number 5665-XA2. It describes and defines the data management macro instructions—except for VSAM (virtual storage access method)—available in the assembler language. See "Related Publications" on page iv for the related VSAM publications.

ORGANIZATION

This publication is divided into these parts:

- "Introduction" on page 1 defines and discusses macro instructions and the rules to be followed when when coding them, and the notational conventions used throughout the publication.
- "Macro Instruction Descriptions" on page 8 defines and discusses the functions of each macro instruction and how each must be coded. The macro instructions are presented in alphabetic order. The standard form of each is described first, followed by the list and execute forms. The list and execute forms are available only for those macro instructions that pass parameters in a list.
- Appendix A, "Status Information Following an Input/Output Operation" on page 192, defines and discusses the location, alignment, and description of the symbolic fields in the data event control block and the event control block.
- Appendix B, "Data Management Macro Instructions Available by Access Method" on page 193, defines and discusses the macro instructions available for each of the data management access methods.
- Appendix C, "Device Capacities" on page 194, defines and discusses device capacities as a guide to coding the block size and logical record length operands in the DCB macro instruction.
- Appendix D, "DCB Exit List Format and Contents" on page 197, defines and discusses the format and content of the data control block exit list.
- Appendix E, "Control Characters" on page 199, defines and discusses the control characters used to control spacing and skipping (printers) and stacker selection (card read punch or card punch).
- Appendix F, "Data Control Block Symbolic Field Names" on page 202, defines and discusses the location, alignment, and description of the data control block symbolic field names.
- Appendix G, "PDABD Symbolic Field Names" on page 219, defines and discusses the location, alignment, and description of the PDABD dummy control section.
- "Glossary of Terms and Abbreviations" on page 220 defines and discusses the definitions of the terms and abbreviations used in this publication.

PREREQUISITE KNOWLEDGE

To use this book you must have knowledge of assembler language as described in Assembler H Version 2 Application Programming: Language Reference, GC26-4037, and Assembler H Version 2 Application Programming: Guide, SC26-4036, and of job control language (JCL) as explained in MVS/Extended Architecture JCL User's Guide, GC28-1351, and MVS/Extended Architecture JCL Reference, GC28-1352.

If you know how to write assembler language programs and use job control statements, you can use this book and MVS/Extended Architecture Data Administration Guide, GC26-4140, to write programs that create and process data sets.

REQUIRED PUBLICATIONS

You should be familiar with the information presented in the following publications:

- Assembler H Version 2 Application Programming: Language Reference, GC26-4037
- Assembler H Version 2 Application Programming: Guide, SC26-4036
- MVS/Extended Architecture Data Administration Guide, GC26-4140
- MVS/Extended Architecture System Programming Library: Supervisor Services and Macro Instructions, GC28-1154
- MVS/Extended Architecture JCL User's Guide, GC28-1351
- MVS/Extended Architecture JCL Reference, GC28-1352

RELATED PUBLICATIONS

Within the text, references are made to the publications listed in the table below:

Short Title	Publication Title	Order Number
Assembler H V2 Application Programming: Guide	Assembler H Version 2 Application Programming: Guide	SC26-4036
Assembler H V2 Application Programming: Language Reference	Assembler H Version 2 Application Programming: Language Reference	GC26-4037
Checkpoint/Restart User's Guide	MVS/Extended Architecture Checkpoint/Restart User's Guide	GC26-4139
Conversion Notebook	MVS/Extended Architecture Conversion Notebook	GC28-1143
Data Administration Guide	MVS/Extended Architecture Data Administration Guide	GC26-4140

Short Title	Publication Title	Order Number
Data Facility Product: Customization	MVS/Extended Architecture Data Facility Product: Version 2 Customization	GC26-4267
IBM 3262 Model 5 Printer Product Description	IBM 3262 Model 5 Printer Product Description	GA24-3936
IBM 3800 Printing Subsystem Programmer's Guide	IBM 3800 Printing Subsystem Programmer's Guide	GC26-3846
IBM 3800 Printing Subsystem Programmer's Guide	IBM 3800 Printing Subsystem Models 3 and 8 Programmer's Guide	SH35-0061
IBM 3890 Document Processor Machine and Programming Description	IBM 3890 Document Processor Machine and Programming Description	GA24-3612
IBM 4245 Printer Model 1 Component Description and Operator's Guide	IBM 4245 Printer Model 1 Component Description and Operator's Guide	GA33-1541
IBM 4248 Printer Model 1 Description	IBM 4248 Printer Model 1 Description	GA24-3927
Initialization and Tuning	MVS/Extended Architecture System Programming Library: Initialization and Tuning	GC28-1149
JCL User's Guide	MVS/Extended Architecture JCL User's Guide	GC28-1351
JCL Reference	MVS/Extended Architecture JCL Reference	GC28-1352
Magnetic Tape Labels and File Structure Administration	MVS/Extended Architecture Magnetic Tape Labels and File Structure Administration	GC26-4145
Open/Close/EOV Logic	MVS/Extended Architecture Open/Close/EOV Logic	LY26-3966
OS/VS IBM 3886 Optical Character Reader Model 1 Reference	OS/VS IBM 3886 Optical Character Reader Model 1 Reference	GC24-5101
OS/VS Mass Storage System (MSS) Planning Guide	OS/VS Mass Storage System (MSS) Planning Guide	GC35-0011
OS/VS Mass Storage System (MSS) Services: General Information	OS/VS Mass Storage System (MSS) Services: General Information	GC35-0016

Short Title	Publication Title	Order Number
OS/VS Mass Storage System (MSS) Extensions Services: Reference	OS/VS Mass Storage System (MSS) Extensions Services: Reference	SH35-0036
Programming Support for the IBM 3505 Card Reader and the IBM 3525 Card Punch	Programming Support for the IBM 3505 Card Reader and the IBM 3525 Card Punch	GC21-5097
RACF General Information Manual	Resource Access Control Facility (RACF): General Information Manual	GC28-0722
Service Aids	MVS/Extended Architecture System Programming Library: Service Aids	GC28-1159
Supervisor Services and Macro Instructions	MVS/Extended Architecture System Programming Library: Supervisor Services and Macro Instructions	GC28-1154
System Codes	MVS/Extended Architecture Message Library: System Codes	GC28-1157
System-Data Administration	MVS/Extended Architecture System-Data Administration	GC26-4149
System Generation	MVS/Extended Architecture Installation: System Generation	GC26-4148
System Macros and Facilities	MVS/Extended Architecture System Programming Library: Macros and Facilities, Volumes 1 and 2	GC28-1150 and GC28-1151
System Messages	MVS/Extended Architecture Message Library: System Messages, Volumes 1 and 2	GC28-1376 and GC28-1377
Utilities	MVS/Extended Architecture Data Administration: Utilities	GC26-4150
VSAM Administration Guide	MVS/Extended Architecture VSAM Administration Guide	GC26-4151
VSAM Administration: Macro Instruction Reference	MVS/Extended Architecture VSAM Administration: Macro Instruction Reference	GC26-4152

SUMMARY OF CHANGES

RELEASE 3.0, JUNE 1987

NEW PROGRAMMING SUPPORT

This release provides virtual storage constraint relief by allowing VSAM control blocks to be obtained above 16M. Changes have been made to each of the three forms of the OPEN and CLOSE macros to allow for a new long form parameter list. The new long form allows for 31-bit addresses. The default is the standard form parameter list with 24-bit addresses.

A new DCB exit list code, X'13', has been added to RDJFCB to retrieve allocation information. Changes have been made to Appendix D, "DCB Exit List Format and Contents" to document the new exit list code.

CUSTOMIZATION RESTRUCTURE

Detailed information on exception codes and status indicators from Appendix A, "Status Information Following an Input/Output Operation" on page 192 has been moved to Data Facility Product: Customization.

SERVICE CHANGES

Minor technical changes have been made. In addition, documentation for the TYPE=J operand has been added to the descriptions of each of the three forms of the OPEN macro.

RELEASE 2.0, AUGUST 1986

Technical changes have been made concerning certain functions. These changes are reflected in sections of the book with vertical change bars.

RELEASE 1.0, APRIL 1985

NEW DEVICE SUPPORT

Information has been added to support the:

- IBM 3380 Models AD4, BD4, AE4, and BE4 Direct Access Storage devices
- IBM 4245, 4248, and 3262 Model 5 Printers
- IBM 3430 Tape Drive
- IBM 3480 Magnetic Tape Subsystem

NEW PROGRAMMING SUPPORT

Information has been added that describes the:

- SYNCDEV macro
- MSGDISP macro

VERSION 2 PUBLICATIONS

The Preface includes new order numbers for Version 2.

CONTENTS

FIGURES

INTRODUCTION

Before using this publication, familiarize yourself with the information in Data Administration Guide.

IBM provides a set of macro instructions so that you can communicate service requests to the data management access method routines. These macro instructions are available only when the assembler language is being used, and they are processed by the assembler program using macro definitions supplied by IBM and placed in the macro library when the operating system is generated.

The assembler program expands each macro instruction into executable, assembler language instructions or data, and shows you the exact macro expansion in the assembler listing. The executable instructions generally consist of branches around data fields, load register instructions, and either branch instructions or supervisor calls (SVC) that transfer control to the proper program. The data fields in each macro instruction are parameters that are passed to the access method routine.

You can use the utility program IEBPTPCH to get a list of macro definitions from SYS1.MACLIB, the library in which the macro definitions are stored. For a description of IEBPTPCH, see Utilities.

Before coding programs that request supervisor services, familiarize yourself with the information contained in Supervisor Services and Macro Instructions. Also, if you are writing programs for specialized applications such as telecommunications, graphics, character recognition, or to use VSAM (virtual storage access method), use the appropriate publication listed in the preface that describes the specific access method and/or device type you are working with.

The operation of some macro instructions depends on the options selected when the macro instruction is coded. For these macro instructions, either separate descriptions are provided or the differences are listed within a single description. If no differences are explicitly listed, none exist. The description of each macro instruction starts on a right-hand page; the descriptions that do not apply to the access methods being used can be removed. Appendix B, "Data Management Macro Instructions Available by Access Method" on page 193 lists the macro instructions available for each access method.

ISO, ANSI, AND FIPS LABELS

This publication refers to tape labels defined by the International Organization for Standardization (ISO), the American National Standards Institute (ANSI), and the Federal Information Processing Standard (FIPS). In general, ISO/ANSI/FIPS labels are similar to IBM standard labels, and, unless otherwise specified, the term "standard label," refers to both IBM standard labels and ISO/ANSI/FIPS standard labels. ISO labeled tapes are coded in the Organization Standard Code for Information Interchange (ISCII), and ANSI labeled tapes are coded in the American National Standards Code for Information Interchange (ASCII), while IBM labeled tapes are coded either in the extended binary-coded-decimal interchange code (EBCDIC) or in binary coded decimal (BCD). For further information about ISO/ANSI/FIPS labels, see Magnetic Tape Labels and File Structure.

CODING AIDS

BOLD TYPE

Bold type is used for elements that you must code exactly as they are shown. These elements consist of macro names, keywords, and these punctuation symbols: commas, parentheses, and equal signs. Examples:

- **DCB**
- **CLOSE ,,,,TYPE=T**
- **MACRF=(PL,PTC)**
- **SK,5**

UNDERSCORED LOWERCASE LETTERS

Underscored lowercase letters are used for elements for which you code values that you choose, usually according to specifications and limits described for each parameter. Examples:

- number
- image-id
- count

BRACKETS

Brackets, [ ], enclose optional elements that you may or may not code as you choose. Examples:

- [length]
- [**MF=E**]

OR SIGN

The OR sign, |, separates alternative elements. Examples:

- [**,REREAD**|**,LEAVE**]
- [length|**'S'**]

BRACES

Braces, { }, enclose alternative elements from which you must choose one, and only one, element. Examples:

- **BFTEK=**{**S**|**A**}
- {**K**|**D**}
- {address|**S**|**0**}

Sometimes, alternative elements (especially complicated alternatives) are grouped in a vertical stack of braces. Example:

```
MACRF={{(R[C|P])}
  {(W[C|P|L])}
  {(R[C],W[C])}}
```

In the examples above, you must choose only one element from the vertical stack.

ELLIPSES

Ellipses, ..., indicate that elements may be repeated.

Example:

- (dcbaddr,[(options)],...)

UNDERSCORED BOLD

Underscored **BOLD** elements indicate alternative choices that are assumed if you don't want to code the optional element. Examples:

- **[EROPT={ACC|SKP|ABE}]**
- **[BFALN={F|D}]**

BLANK SYMBOL

The blank symbol, b, indicates omitted operands. Example:

b	**PDAB**	b

COMPREHENSIVE EXAMPLE

- **MF=(E,{**address**|(1)})**

  In this example, **MF=(E,** must be coded exactly as shown. Then, either address or **(1)** must be coded. (The parentheses around the **1** are required.) Finally, the closing parenthesis must be coded. Thus, **MF=(E,(1))** might be coded.

- **RECFM={{U[T][A|M]}**
  {V[B|S|T|BS|BT][A|M]}
  {D[B][A]}
  {F[B|S|T|BS|BT][A|M]}}

  In this example, you must first choose one of the four alternative elements shown on each line. Then, you must choose one of the major elements. Assuming you selected the major element beginning with F, you would code **F**; then you could choose one of **B**, **S**, **T**, **BS**, or **BT**. Finally, you could select either **A** or **M**. Thus, you might code any one of the following: **RECFM=FBTM**, **RECFM=FA**, or **RECFM=F**.

MACRO INSTRUCTION FORMAT

Data management macro instructions are subject to the rules of assembler language and are written in the following format:

Name	Operation	Operands	Comments
Symbol or blank	Macro name	None, one or more operands separated by commas	

Use the operands to specify services and options you need and code them according to the following general rules:

- If the operand you select is shown in bold capital letters (for example, **MACRF=WL**), code the operand exactly as shown.

- If the operand you select is a character string in bold type (for example, if the type operand of a READ macro instruction is **SF**), code the operand exactly as shown.

- If the operand is shown in underscored lowercase letters (for example, dcb address), substitute the indicated address, name, or value.

- If the operand is a combination of bold capital letters and underscored lowercase letters (for example, **LRECL=**absexp), code the capital letters and equal sign exactly as shown and substitute the appropriate address, name, or value for the underscored lowercase letters.

- Code commas and parentheses exactly as shown.

  Note: Omit the comma that follows the last operand in a statement. Brackets and braces show how to use commas and parentheses the same way they show how to use operands.

- Several macro instructions contain the designation **'S'**. Use the apostrophe on both sides of the **S** operand.

If you need to substitute a name, value, or address, the notation you use depends on the operand you are coding. The following two examples show how an operand can be coded:

DDNAME=symbol
In this example, you can only code a valid assembler-language symbol for the operand.

dcb address—RX-Type Address, (2-12), or (1)
In the above example, you can substitute an RX-type address, any general register 2 through 12, or general register 1.

The following examples show what each notation means and how you can code an operand:

symbol
This notation indicates that the operand can be any valid assembler-language symbol.

decimal digit
This notation indicates that the operand can be any decimal digit up to the maximum value allowed for the specific operand.

(2-12)
This notation indicates that the operand can be any of the general registers 2 through 12. All register operands must be coded in parentheses; for example, if you code register 3, use the form (3). If you want to use one of the registers 2 through 12, code it as a decimal digit, a symbol (equated to a decimal digit), or an expression that yields a value of 2 through 12.

(1)
When this notation is shown, general register 1 can be used as an operand. The register can be specified as a decimal digit 1 enclosed in parentheses. When register 1 is used as an operand, the instruction that loads the parameter value into the register is not included in the macro expansion.

(0)
When this notation is shown, general register 0 can be used as an operand. The register can be specified as a decimal

digit <u>0</u> enclosed in parentheses. When register <u>0</u> is used as an operand, the instruction that loads the parameter value into the register is not included in the macro expansion.

<u>RX-Type Address</u>
When this notation is shown, the operand can be specified as any valid assembler-language RX-type address. The following shows examples of each valid RX-type address:

Name	**Operation**	**Operand**
ALPHA1	L	1,39(4,10)
ALPHA2	L	REG1,39(4,TEN)
BETA1	L	2,ZETA(4)
BETA2	L	REG2,ZETA(REG4)
GAMMA1	L	2,ZETA
GAMMA2	L	REG2,ZETA
GAMMA3	L	2,=F'1000'
LAMBDA1	L	3,20(,5)

Both ALPHA instructions specify explicit addresses; REG1 and TEN are absolute symbols. Both BETA instructions specify implied addresses, and both use index registers. Indexing is omitted from the GAMMA instructions. GAMMA1 and GAMMA2 specify implied addresses. The second operand of GAMMA3 is a literal. LAMBDA1 specifies an explicit address with no indexing.

<u>A-Type Address</u>
When this notation is shown, the operand can be specified as any address that can be written as a valid assembler-language A-type address constant. An A-type address constant can be written as an absolute value, a relocatable symbol, or a relocatable expression. Operands that require an A-type address are inserted into an A-type address constant during the macro expansion process. For more details about A-type address constants, see <u>Assembler H Version 2 Application Programming: Language Reference</u>.

<u>absexp</u>
When this notation is shown, the operand can be an absolute value or expression. An absolute expression can be an absolute term or an arithmetic combination of absolute terms. An absolute term can be a nonrelocatable symbol, a self-defining term, or the length attribute reference. For more details about absolute expressions, see <u>Assembler H Version 2 Application Programming: Language Reference</u>.

<u>relexp</u>
When this notation is shown, the operand can be a relocatable symbol or expression. A relocatable symbol or expression is one whose value changes by n if the program in which it appears is relocated n bytes away from its originally assigned area of storage. For more details about relocatable symbols and expressions, see <u>Assembler H Version 2 Application Programming: Language Reference</u>.

RULES FOR REGISTER USAGE

Many macro instruction expansions include instructions that use a base register previously defined by a USING statement. The USING statement must establish addressability so that macro expansion can include a branch around the in-line parameter list, if present, and see the data fields and addresses specified in the macro instruction operands.

Macro instructions that use a BAL or BALR instruction to pass control to an access method routine, normally require that register 13 contain the address of an 18-word register-save area. The READ, WRITE, CHECK, GET, and PUT macro instructions are of this type.

Macro instructions that use a supervisor call (SVC) instruction to pass control to an access method routine may modify general registers 0, 1, 14, and 15 without restoring them. Unless otherwise specified in the macro instruction description, the contents of these registers are undefined when the system returns control to the problem program.

When an operand is specified as a register, the problem program must have inserted the value or address to be used into the register as follows:

- Unless the macro instruction description states otherwise, and the register is to contain a value, that value must be placed in the low-order portion of the register. Any unused bits in the register should be set to zero.
- If the register is to contain a 21-bit address, the address must be placed in the low-order three bytes of the register, and the high-order byte of the register should be set to zero.
- If the register is to contain a 31-bit address, the address must be placed in the low-order 31 bits of the register, and the high-order bit of the register should be set to zero.

Note that, if the macro instruction accepts the RX-type address, the high-order byte of a register can be efficiently cleared by coding the parameter as 0 (reg) rather than merely as (reg).[1] Then the macro instruction expands as:

```
  LA   parmreg,0(reg)      by macro
```

rather than:

```
  LA   reg,0(reg)          by user
```

and

```
  LR   parmreg,reg         by macro
```

RULES FOR CONTINUATION LINES

The operand field of a macro instruction can be continued on one or more additional lines as follows:

1. Enter a continuation character (not blank, and not part of the operand coding) in column 72 of the line.
2. Continue the operand field on the next line, starting in column 16. All columns to the left of column 16 must be blank.

---

[1] For 31-bit addressing mode expansion, the high-order bit of a register can be cleared using this same technique.

The operand field being continued can be coded in one of two ways. The operand field can be coded through column 71, with no blanks, and be continued in column 16 of the next line, or the operand field can be truncated by a comma, where a comma normally falls, with at least one blank before column 71, and then be continued in column 16 of the next line. An example of each method is shown in the following illustration:

```
Name  Operation  Operand                                Comments

NAME1  OP1        OPERAND1,OPERAND2,OPERAND3,OPERAND4,OPERAND5,OPERAND6,OPERX
                  AND7,OPERAND8              THIS IS ONE WAY

NAME2  OP2        OPERAND1,OPERAND2,         THIS IS ANOTHER WAY            X
                  OPERAND3,                                                 X
                  OPERAND4
```

BLDL—BUILD A DIRECTORY ENTRY LIST (BPAM)

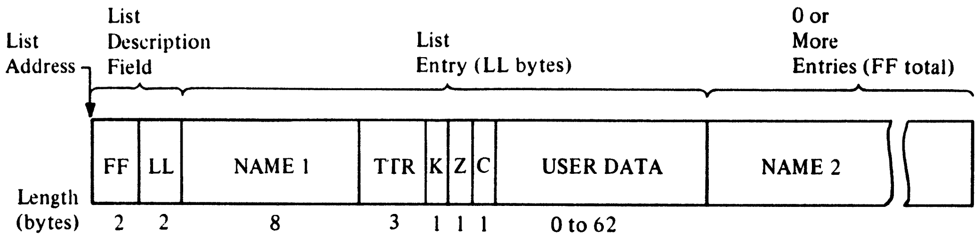
The BLDL macro is used to complete a list of information from the directory of a partitioned data set. The problem program must supply a storage area that must include information about the number of entries in the list, the length of each entry, and the name of each data set member (or alias) before the BLDL macro is issued. Data set member names in the list must be in alphameric order. All read and write operations using the same data control block must have been tested for completion before the BLDL macro is issued.

The BLDL macro is written:

[symbol]	**BLDL**	dcb address ,list address

dcb address—RX-Type Address, (2-12) or (1)
The dcb address operand specifies the address of the data control block for an open partitioned data set, or zero can be specified to indicate that the data set is in a job library, step library, or link library.

list address—RX-Type Address, (2-12), or (0)
The list address operand specifies the address of the list to be completed when the BLDL macro is issued. The list address must be on a halfword boundary. The following illustration shows the format of the list:



FF: This field must contain a binary value indicating the total number of entries in the list.

LL: This field must contain a binary value indicating the length, in bytes, of each entry in the list (must be an even number of bytes). If the exact length of the entry is known, specify the exact length. Otherwise, specify at least 58 bytes (decimal) if the list is to be used with an ATTACH, LINK, LOAD, or XCTL macro. The minimum length for a list is 12 bytes.

NAME: This field must contain the member name or alias to be located. The name must start in the first byte of the name field and be padded to the right with blanks (if necessary) to fill the 8-byte field.

When the BLDL macro is executed, five fields of the directory entry list are filled in by the system. The specified length (LL) must be at least 14 to fill in the **Z** and **C** fields. If the **LL** field is 12, only the **NAME, TT, R,** and K fields are returned. The five fields are:

TT: Indicates the relative track number where the beginning of the data set member is located.

R: Indicates the relative block (record) number on the track indicated by **TT**.

K: Indicates the concatenation number of the data set. For the first or only data set, this value is zero.

Z: Indicates where the system found the directory entry:

Code	**Meaning**
0	Private library
1	Link library
2	Job, task, or step library
3-255	Job, task, or step library of parent task n, where n = Z-2

C: Indicates the type (member or alias) for the name, the number of note list fields (TTRNs), and the length of the user data field (indicated in halfwords). The following describes the meaning of the 8 bits:

Bit	**Meaning**
0=0	Indicates a member name.
0=1	Indicates an alias.
1-2	Indicate the number of TTRN fields (maximum of 3) in the user data field.
3-7	Indicate the total number of halfwords in the user data field. If the list entry is to be used with an ATTACH, LINK, LOAD, or XCTL macro, the value in bits 3 through 7 is 22 (decimal).

USER DATA: The user data field contains the user data from the directory entry. If the length of the user data field in the BLDL list is equal to or greater than the user data field of the directory entry, the entire user data field is entered into the list. Otherwise, the list contains only the user data for which there is space.

COMPLETION CODES

When the system returns control to the problem program, the low-order byte of register 15 contains a return code; the low-order byte of register 0 contains a reason code, as follows:

Return Code (15)	Reason Code (0)	Meaning
00 (X'00')	00 (X'00')	Successful completion.
04 (X'04')	00 (X'00')	One or more entries in the list could not be filled; the list supplied may be invalid. If a search is attempted but the entry is not found, the R field (byte 11) for that entry is set to zero.
08 (X'08')	00 (X'00')	A permanent I/O error was detected when the system attempted to search the directory.
08 (X'08')	04 (X'04')	Insufficient virtual storage was available.
08 (X'08')	08 (X'08')	Invalid DEB. (Not in key 0 through 7.)

BSP—BACKSPACE A PHYSICAL RECORD (BSAM—MAGNETIC TAPE AND DIRECT ACCESS ONLY)

The BSP macro causes the current volume to be backspaced one data block (physical record). All input and output operations must be tested for completion before the BSP macro is issued. The BSP macro should not be used if the CNTRL, NOTE, or POINT macro is being used. The BSP macro can be used only on data sets created by BSAM.

Any attempt to backspace across a file mark will result in a return code of X'04' and your tape or direct access volume will not be repositioned. This means you cannot issue a successful BSP macro after your EODAD routine is entered unless you first reposition the tape or direct access volume into your data set. (CLOSE TYPE=T would get you repositioned at the end of your data set.)

Magnetic Tape: A backspace is always made toward the beginning of the tape.

Direct Access Device: A BSP macro must not be issued for a data set created by using track overflow.

SYSIN or SYSOUT Data Sets: A BSP macro is ignored, but a completion code is returned.

The BSP macro is written:

[symbol]	**BSP**	dcb address

dcb address—RX-Type Address, (2-12), or (1)
The dcb address operand specifies the address of the data control block for the volume to be backspaced. The data set on the volume to be backspaced must be opened before issuing the BSP macro.

COMPLETION CODES

When the system returns control to the problem program, the low-order byte of register 15 contains a return code; the low-order byte of register 0 contains a reason code, as follows:

Return Code (15)	Reason Code (0)	Meaning
00 (X'00')	00 (X'00')	Successful completion.
04 (X'04')	01 (X'01')	A backspacing request was ignored on a SYSIN or SYSOUT data set.
04 (X'04')	02 (X'02')	Backspace not supported for this device type.
04 (X'04')	03 (X'03')	Backspace not successful; insufficient virtual storage was available.
04 (X'04')	04 (X'04')	Backspace not successful; permanent I/O error.
04 (X'04')	05 (X'05')	Backspace into load point or beyond start of data set on the current volume.
04 (X'04')	06 (X'06')	Backspace detected an invalid DEB using DEBCHECK.
04 (X'04')	07 (X'07')	Backspace detected an invalid extend value (M).
04 (X'04')	08 (X'08')	Backspace issued while I/O was in progress.

BUILD—BUILD A BUFFER POOL (BDAM, BISAM, BPAM, BSAM, QISAM, AND QSAM)

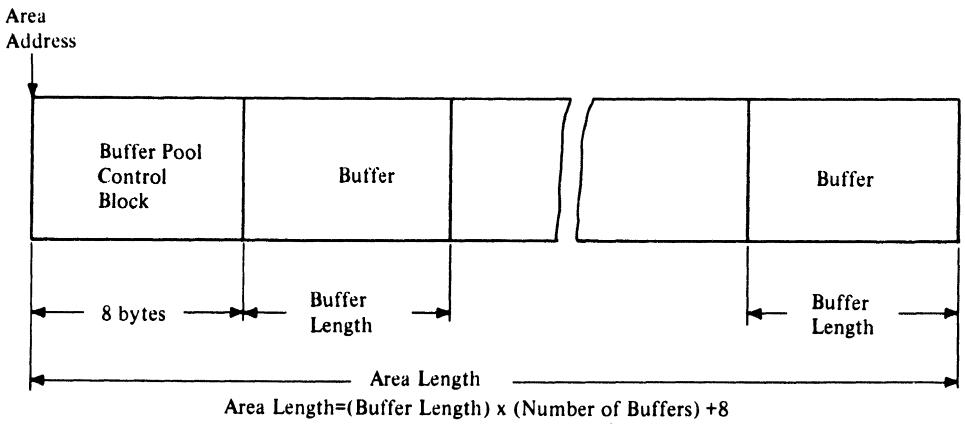
The BUILD macro is used to construct a buffer pool in an area provided by the problem program. The buffer pool may be used by more than one data set through separate data control blocks. Individual buffers are obtained from the buffer pool using the GETBUF macro, and buffers are returned to the buffer pool using a FREEBUF macro. See Data Administration Guide for an explanation of the interaction of the DCB, BUILD, and GETBUF macros in each access method, and the buffer size requirements.

The BUILD macro is written:

[symbol]	**BUILD**	area address ,{number of buffers,buffer length|(0)}

area address—RX-Type Address, (2-12), or (1)
The area address operand specifies the address of the area to be used as a buffer pool. The area must start on a fullword boundary.

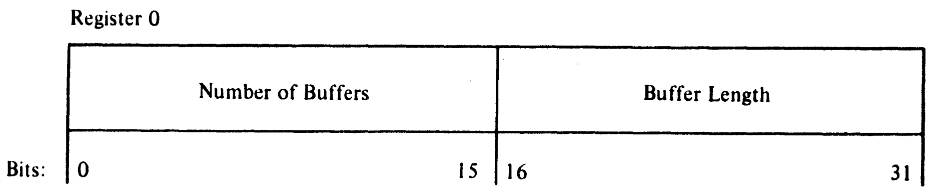
The following illustration shows the format of the buffer pool:



number of buffers—symbol, decimal digit, absexp, or (2-12)
The number-of-buffers operand specifies the number of buffers in the buffer pool to a maximum of 255.

buffer length—symbol, decimal digit, absexp, or (2-12)
The buffer length operand specifies the length, in bytes, of each buffer in the buffer pool. The value specified for the buffer length must be a fullword multiple; otherwise, the system rounds the value specified to the next higher fullword multiple. The maximum length that can be specified is 32760 bytes. For QSAM, the buffer length must be at least as large as the value specified in the block size (DCBBLKSI) field of the data control block.

(0)
The number of buffers and buffer length can be specified in general register 0. If (0) is coded, register 0 must contain the binary values for the number of buffers and buffer length as shown in the following illustration.



BUILDRCD—BUILD A BUFFER POOL AND A RECORD AREA (QSAM)

The BUILDRCD macro causes a buffer pool and a record area to be constructed in a user-provided storage area. This macro is used only for variable-length, spanned records processed in QSAM locate mode. If the extended logical record interface (XLRI) is used to process RECFM=DS or RECFM=DBS records (ISO/ANSI/FIPS variable spanned or variable blocked spanned), you can use the BUILDRCD macro to build a record area to a maximum length of 16777183 bytes. Use of this macro before the data set is opened, or before the end of the DCB open exit routine, will provide a buffer pool that can be used for a logical record interface rather than a segment interface for variable-length spanned records. To invoke a logical record interface, specify **BFTEK=A** in the DCB. The **BUILDRCD** macro cannot be specified when logical records exceed 32760 bytes.

The standard form of the BUILDRCD macro is written as follows (the list and execute forms are shown following the description of the standard form):

[symbol]	**BUILDRCD**	area address ,number of buffers ,buffer length ,record area address [,record area length]

area address—A-Type Address or (2-12)
The area address operand specifies the address of the area to be used as a buffer pool. The area must start on a fullword boundary.

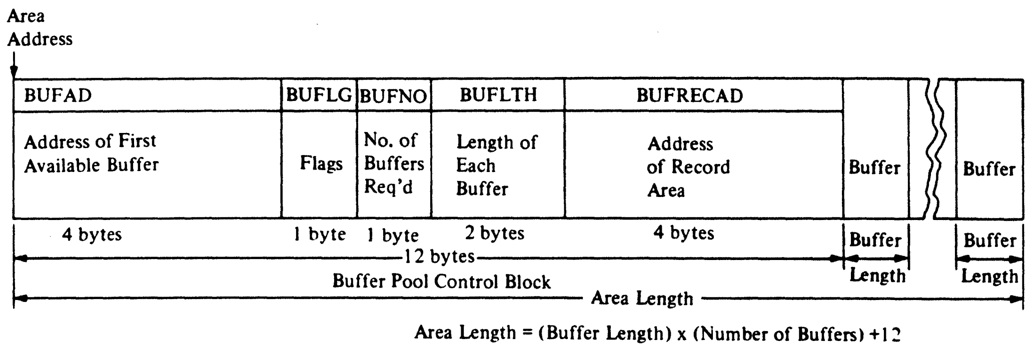
number of buffers—symbol, decimal digit, absexp, or (2-12)
The number-of-buffers operand specifies the number of buffers, to a maximum of 255, to be in the buffer pool.

buffer length—symbol, decimal digit, absexp, or (2-12)
The buffer-length operand specifies the length, in bytes, of each buffer in the buffer pool. The value specified for the buffer length must be a fullword multiple; otherwise, the system rounds the value specified to the next higher fullword multiple. The maximum length that can be specified is 32760.

record area address—A-Type Address or (2-12)
The record area address operand specifies the address of the storage area to be used as a record area. The area must start on a doubleword boundary and have a length of the maximum logical record (LRECL) plus 32 bytes.

record area length—symbol, decimal digit, absexp, or (2-12)
The record area length operand specifies the length of the record area to be used. The area must be as long as the maximum length logical record plus 32 bytes for control information. If the record area length operand is omitted, the problem program must store the record area length in the first four bytes of the record area.

The following illustration shows the format of the buffer pool:



BUFLG Flags:

Bit	Meaning
0=1	Record area present
1=1	Buffer control block extended
2-7	Reserved

Notes:

1. The buffer pool control block contains the address of the record area and a flag that indicates logical-record interface processing of variable-length, spanned records.

2. It is the user's responsibility to release the buffer pool and the record area after a CLOSE macro has been issued for all the data control blocks that use the buffer pool and the record area.

BUILDRCD—LIST FORM

The list form of the BUILDRCD macro is used to construct a program parameter list. The description of the standard form of the BUILDRCD macro instruction provides the explanation of the function of each operand. The description of the standard form also indicates the operands that are totally optional and those that are required in at least one of the pair of list and execute forms. The format description below indicates the optional and required operands in the list form only.

The list form of the BUILDRCD macro is written:

[symbol]	**BUILDRCD**	area address ,number of buffers ,buffer length ,record area address [,record area length] ,**MF=L**

area address—A-Type Address

number of buffers—symbol, decimal digit, or absexp

buffer length—symbol, decimal digit, or absexp

record area address—A-Type Address

record area length—symbol, decimal digit, or absexp

MF=L
: The **MF=L** operand specifies that the BUILDRCD macro instruction is used to create a parameter list that will be referenced by an execute form instruction.

Note: A parameter list can be constructed by coding only the **MF=L** operand (without the preceding comma); in this case, the list is constructed for the area address, number of buffers, buffer length, and record area address operands. If the record area length operand is also required, the operands can be coded as follows:

```
    [symbol] BUILDRCD ,,,0,MF=L
```

The preceding example shows the coding to construct a list containing address constants with a value of 0 in each constant. The actual values can then be supplied by the execute form of the BUILDRCD macro.

BUILDRCD—EXECUTE FORM

A remote parameter list is referred to, and can be modified by, the execute form of the BUILDRCD macro. The description of the standard form of the BUILDRCD macro instruction provides the explanation of the function of each operand. The description of the standard form also indicates the operands that are totally optional and those that are required in at least one of the pair of list and execute forms. The format description below indicates the optional and required operands for the execute form only.

The execute form of the BUILDRCD macro is written:

[symbol]	**BUILDRCD**	[area address] ,[number of buffers] ,[buffer length] ,[record area address] ,[record area length] ,MF=(E,{list address|(1)})

area address—RX-Type Address or (2-12)

number of buffers—absexp

buffer length—absexp

record area address—RX-Type Address or (2-12)

record area length—absexp

MF=(E,{list address|(1)})
This operand specifies that the execute form of the BUILDRCD macro instruction is used, and an existing parameter list (created by a list-form instruction) will be used. The MF= operand is coded as described in the following:

E

list address—RX-Type Address, (2-12), or (1)

CHECK—WAIT FOR AND TEST COMPLETION OF A READ OR WRITE OPERATION (BDAM, BISAM, BPAM, AND BSAM)

The CHECK macro causes the active task to be placed in the wait condition, if necessary, until the associated input or output operation is completed. The input or output operation is then tested for errors and exceptional conditions. If the operation is completed successfully, control is returned to the instruction following the CHECK macro. If the operation is not completed successfully, the error analysis (SYNAD) routine is given control or, if no error analysis routine is provided, the task is abnormally terminated. The error analysis routine is discussed in the SYNAD operand of the DCB macro.

The following conditions are also handled for BPAM and BSAM only:

When Reading: The end-of-data (EODAD) routine is given control if an input request is made after all the records have been retrieved. Volume switching is automatic for a BSAM data set that is not opened for UPDAT. For a BSAM data set that is opened for update, the end-of-data routine is entered at the end of each volume.

When Writing: Additional space on the device is obtained when the current space is filled and more WRITE macro instructions have been issued.

For BPAM and BSAM, a CHECK macro must be issued for each input and output operation, and must be issued in the same order as the READ or WRITE macros were issued for the data set. For BDAM or BISAM, either a CHECK or a WAIT macro can be used. For information on when the **WAIT** macro can be used, see _Data Administration Guide_.

If the ISCII/ASCII translation routines are included when the operating system is generated, translation can be requested by coding LABEL=(,AL) or (,AUL) in the DD statement, or it can be requested by coding OPTCD=Q in the DCB macro or DCB subparameter of the DD statement. If translation is requested, the check routine automatically translates BSAM records, as they are read, from ISCII/ASCII code to EBCDIC code, provided that the record format is F, FB, D, DB, or U. Translation occurs as soon as the check routine determines that the input buffer is full. For translation to occur correctly, all input data must be in ISCII or ASCII code.

The CHECK macro is written:

[_symbol_]	**CHECK**	_decb address_ **[,DSORG={IS|ALL}]**

decb address—RX-Type Address, (2-12), or (1)
The decb address operand specifies the address of the data event control block created by the associated READ or WRITE macro or used by the associated input or output operation.

DSORG={IS|ALL}
The **DSORG** operand specifies the type of data set organization. The following describes the characters that can be coded:

IS
Specifies that the program generated is for BISAM use only.

ALL
Specifies that the program generated is for BDAM, BISAM, BPAM, or BSAM use.

If the **DSORG** operand is omitted, the program generated is for BDAM, BPAM, or BSAM use only.

CHKPT—TAKE A CHECKPOINT FOR RESTART WITHIN A JOB STEP

The CHKPT macro is coded in-line in the problem program. When this macro executes, the operating system writes a checkpoint entry in a checkpoint data set. The entry consists of job step information, such as virtual-storage data areas, data set position, and supervisor control, from the problem program. The problem program automatically restarts with the instruction immediately following the CHKPT macro.

For details on the CHKPT macro, see Checkpoint/Restart User's Guide.

CLOSE—LOGICALLY DISCONNECT A DATA SET (BDAM, BISAM, BPAM, BSAM, QISAM, AND QSAM)

The CLOSE macro causes output data set labels to be created and volumes to be positioned as specified by the user. The fields of the data control block are restored to the condition that existed before the OPEN macro was issued, and the data set is disconnected from the processing program. Final volume positioning or disposition for the current volume can be specified to override the positioning implied by the DD control statement DISP parameter. Any number of dcb address operands and associated options may be specified in the CLOSE macro.

Associated data sets for an IBM 3525 Card Punch can be closed in any sequence, but, if one data set is closed, I/O operations cannot be initiated for any of its associated data sets. Additional information about closing associated data sets is contained in Data Administration Guide.

After a CLOSE has been issued for several data sets, a return code of 4 indicates that at least one of the data sets, VSAM or non-VSAM, was not closed successfully.

A FREEPOOL macro should normally follow a CLOSE macro instruction (without TYPE=T) to regain the buffer pool storage space and to allow a new buffer pool to be built if the DCB is reopened with different record size attributes.

A special operand, **TYPE=T**, is provided for processing with BSAM.

The standard form of the CLOSE macro is written as follows (the list and execute forms are shown following the description of the standard form):

[symbol]	**CLOSE**	(dcb address,[option,...]) **[,TYPE=T]** **[,MODE=24|31]**

dcb address—A-Type Address or (2-12)
: specifies the address of the data control block for the opened data set that is to be closed.

option
: One of these options indicates the volume positioning that is to occur when the data set is closed. This option is generally used with the **TYPE=T** operand or for data sets on magnetic tape. However, options specified in the CLOSE macro will override disposition specifications in the JCL for all data sets. The options are:

REREAD
: specifies that the current volume is to be positioned to reprocess the data set. If processing was forward, the volume is positioned to the beginning of the data set; if processing was backward (RDBACK), the volume is positioned to the end of the data set. If **FREE=CLOSE** is specified in the JCL, the data set is not unallocated until the end of the job step.

LEAVE
: specifies that the current volume is to be positioned to the logical end of the data set. If processing was forward, the volume is positioned to the end of the data set; if processing was backward (RDBACK), the volume is positioned to the beginning of the data set.

REWIND
: specifies that the current magnetic tape volume is to be positioned at the load point, regardless of the direction of processing. **REWIND** cannot be specified when **TYPE=T** is specified. If **FREE=CLOSE** has been coded on the DD statement associated with the data set being closed, coding the **REWIND** option will result in

the data set being freed at the time it is closed rather than at the termination of the job step.

FREE
specifies that the current data set is to be freed at the time the data set is closed, rather than at the time the job step is terminated. For tape data sets, this means that the volume is eligible for use by other tasks or to be demounted. Direct access volumes may also be freed for use by other tasks. They may be freed for demounting if (1) no other data sets on the volume are open and (2) the volume is otherwise demountable. Do not use this option with CLOSE **TYPE=T**. (For other restrictions on the FREE parameter, see _JCL_.)

DISP
specifies that a tape volume is to be disposed of in the manner implied by the DD statement associated with the data set. Direct access volume positioning and disposition are not affected by this parameter. There are several dispositions that can be specified in the **DISP** parameter of the DD statement; **DISP** can be **PASS, DELETE, KEEP, CATLG,** or **UNCATLG.**

Depending on how the **DISP** option is coded in the DD statement, the current magnetic tape volume will be positioned as follows:

DISP Parameter	Action
PASS	Forward space to the end of data set on the current volume.
DELETE	Rewind the current volume.
KEEP, CATLG, or UNCATLG	The volume is positioned as for CLOSE REREAD. Note that the volume is not rewound and unloaded.

If **FREE=CLOSE** has been coded in the DD statement associated with this data set, coding the **DISP** option in the CLOSE macro will result in the data set being freed when the data set is closed, rather than at the time the job step is terminated.

Note: When the option operand is omitted, **DISP** is assumed. For **TYPE=T**, this is processed as **LEAVE** during execution.

The **LEAVE** and **REREAD** options are meaningless except for magnetic tape and **CLOSE TYPE=T**.

TYPE=T
You can code CLOSE **TYPE=T** to perform some close functions for sequential data sets on magnetic tape and direct access volumes processed with BSAM. When you use **TYPE=T**, the DCB used to process the data set maintains its open status, and you should not issue another OPEN macro to continue processing the same data set. This option cannot be used in a SYNAD exit routine.

The **TYPE=T** operand causes the system control program to process labels, modify some of the fields in the system control blocks for that data set, and reposition the volume (or current volume in the case of multivolume data sets) in much the same way that the normal CLOSE macro does. When you code **TYPE=T**, you can specify that the volume either be positioned at the end of data (the **LEAVE** option) or be repositioned at the beginning of data (the **REREAD** option). Magnetic tape volumes are repositioned either immediately before the first data record or immediately after the last data record; the presence of tape labels has no effect on repositioning.

If you code the RLSE keyword with the SPACE parameter on the DD statement that describes the output data set, it is ignored by temporary close (CLOSE **TYPE=T**). If the last operation occurring prior to the normal CLOSE (without **TYPE=T**) and after the temporary close was a write, then any unused space will be released.

MODE=<u>24</u>|31

You can code CLOSE **MODE=31** to specify a long form parameter list that will be able to contain 31-bit addresses. The default, **MODE=24**, will specify a standard form parameter list with 24-bit addresses.

The standard form parameter list is 4 bytes per entry. The standard form parameter list must reside below 16M, but the calling program may be above 16M. It is assumed that all ACBs and DCBs are below 16M.

The long form parameter list can reside above or below 16M. Each entry is 8 bytes long. Option information is contained in the first byte, zeros in the next three bytes, and the address of the ACB or DCB is contained in the last four bytes. Although the ACB or DCB address is contained in a 4-byte field, the DCB must be below 16M. Except for VSAM or VTAM ACBs, all ACBs must also be below 16M. Therefore, the leading byte of the ACB or DCB address must contain zeros. If the byte contains something other than zeros, an IEC290I message will be issued and the data set will not be closed.

Note: It is up to you to keep the mode specified in the **MF=L** and **MF=E** versions of the OPEN and CLOSE macros consistent. Errors and unpredictable results will occur if the specified modes are inconsistent.

Note: To use the CLOSE macro instruction supplied in the MVS/XA macro library on MVS/370, use the SPLEVEL macro instruction. You must use the SPLEVEL macro instruction to ensure that the MVS/XA version of the CLOSE macro instruction executes successfully on MVS/370. For more information on how to use the SPLEVEL macro, see <u>System Macros and Facilities</u>, Volume 2.

For additional information and coding restrictions, see <u>Data Administration Guide</u>.

CLOSE—LIST FORM

The list form of the CLOSE macro is used to construct a data management parameter list. Any number of operands (data control block addresses and associated options) can be specified.

The list consists of a one-word entry for each DCB in the parameter list; the high-order byte is used for the options and the three low-order bytes are used for the DCB address. The end of the list is indicated by a 1 in the high-order bit of the last entry's option byte. The length of a list generated by a list-form instruction must be equal to the maximum length required by an execute-form instruction that refers to the same list. A maximum length list can be constructed by one of two methods:

- Code a list-form instruction with the maximum number of parameters that are required by an execute-form instruction that refers to the list.
- Code a maximum length list by using commas in a list-form instruction to acquire a list of the appropriate size. For example,, coding CLOSE (,,,,,,,,,),MF=L would provide a list of five fullwords (five dcb addresses and five options).

Entries at the end of the list that are not referenced by the execute-form instruction are assumed to have been filled in when the list was constructed or by a previous execute-form instruction. Before using the execute-form instruction, you may shorten the list by placing a 1 in the high-order bit of the last DCB entry to be processed.

A zeroed work area on a word boundary is equivalent to CLOSE (,DISP,...),MF=L and can be used in place of a list-form instruction. The high-order bit of the last DCB entry must contain a 1 before this list can be used with the execute-form instruction.

A parameter list constructed by a CLOSE macro, list form, can be referred to by either an OPEN or CLOSE execute-form instruction.

The description of the standard form of the CLOSE macro provides the explanation of the function of each operand. The description of the standard form also indicates the operands that are completely optional and those required in at least one of the pair of list and execute forms. The format description below indicates the optional and required operands in the list form only.

The list form of the CLOSE macro is written:

[symbol]	**CLOSE**	([dcb address],[option],...) [,TYPE=T] ,MF=L [,MODE=24|31]

dcb address—A-Type Address

option—Same as standard form

TYPE=T
The **TYPE=T** operand can be coded in the list-form instruction to allow the specified option to be checked for validity when the program is assembled.

MF=L
The **MF=L** operand specifies that the CLOSE macro instruction is used to create a data management parameter list that will be referred to by an execute-form instruction.

MODE=24|31

You can code CLOSE **MODE=31** to specify a long form parameter list that will be able to contain 31-bit addresses. The default, **MODE=24**, will specify a standard form parameter list with 24-bit addresses.

The standard form parameter list is 4 bytes per entry. The standard form parameter list must reside below 16M, but the calling program may be above 16M. It is assumed that all ACBs and DCBs are below 16M.

The long form parameter list can reside above or below 16M. Each entry is 8 bytes long. Option information is contained in the first byte, zeros in the next three bytes, and the address of the ACB or DCB is contained in the last four bytes. Although the ACB or DCB address is contained in a 4-byte field, the DCB must be below 16M. Except for VSAM or VTAM ACBs, all ACBs must also be below 16M. Therefore, the leading byte of the ACB or DCB address must contain zeros. If the byte contains something other than zeros, an IEC290I message will be issued and the data set will not be closed.

Note: It is up to you to keep the mode specified in the **MF=L** and **MF=E** versions of the OPEN and CLOSE macros consistent. Errors and unpredictable results will occur if the specified modes are inconsistent.

CLOSE—EXECUTE FORM

A list form of the CLOSE macro is used in and can be modified by the execute form of the CLOSE macro. The parameter list can be generated by the list form of either an OPEN macro or a CLOSE macro.

The description of the standard form of the CLOSE macro provides the explanation of the function of each operand. The description of the standard form also indicates the operands that are totally optional and those required in at least one of the pair of list and execute forms. The format description below indicates the optional and required operands in the execute form only.

The execute form of the CLOSE macro is written:

[_symbol_]	**CLOSE**	[([_dcb address_],[_option_],...)] [,TYPE=T] ,MF=(E,{_address of list form_|(1)}) [,MODE=_24_|31]

dcb address—RX-Type Address or (2-12)

option—If specified, same as the standard form. If not specified, the option specified in the list form of the CLOSE macro will be used.

TYPE=T—Same as standard form.

MF=(E,{_address of the list form_|(1)})
specifies that the execute form of the CLOSE macro instruction is being used, and the parameter list is created by the list form of the CLOSE macro instruction. The MF= operand is coded as described in the following:

E
address of the list form of the CLOSE (or OPEN) macro instruction—RX-Type Address, (2-12), or (1)

MODE=_24_|31
You can code CLOSE **MODE=31** to specify a long form parameter list that will be able to contain 31-bit addresses. The default, **MODE=24**, will specify a standard form parameter list with 24-bit addresses.

The standard form parameter list is 4 bytes per entry. The standard form parameter list must reside below 16M, but the calling program may be above 16M. It is assumed that all ACBs and DCBs are below 16M.

The long form parameter list can reside above or below 16M. Each entry is 8 bytes long. Option information is contained in the first byte, zeros in the next three bytes, and the address of the ACB or DCB is contained in the last four bytes. Although the ACB or DCB address is contained in a 4-byte field, the DCB must be below 16M. Except for VSAM or VTAM ACBs, all ACBs must also be below 16M. Therefore, the leading byte of the ACB or DCB address must contain zeros. If the byte contains something other than zeros, an IEC290I message will be issued and the data set will not be closed.

Note: It is up to you to keep the mode specified in the **MF=L** and **MF=E** versions of the OPEN and CLOSE macros consistent. Errors and unpredictable results will occur if the specified modes are inconsistent.

CNTRL—CONTROL ON-LINE INPUT/OUTPUT DEVICE (BSAM AND QSAM)

The CNTRL macro is used to control magnetic tape drives (BSAM only for a data set that is not open for output), on-line card readers, IBM 3525 Card Punches (read and print features), printers (BSAM and QSAM), and the IBM 3890 Document Processor (QSAM only). For information on additional operands for the CNTRL macro for the 3890, see IBM 3890 Document Processor Machine and Programming Description.

The MACRF operand of the DCB macro must specify a C. The CNTRL macro is ignored for SYSIN or SYSOUT data sets. For BSAM, all input and output operations must be tested for completion before the CNTRL macro is issued. The control facilities available are as follows:

Card Reader: Provides stacker selection, as follows:

QSAM—For unblocked records, a CNTRL macro should be issued after every input request. For blocked records, a CNTRL macro is issued after the last logical record on each card that is retrieved. Whether reading blocked or unblocked records, do not issue a CNTRL macro after a GET macro has caused control to pass to the EODAD routine. The move mode of the GET macro must be used, and the number of buffers (BUFNO field of the DCB) must be 1. If a CLOSE macro is issued before the last card is read, the operator should clear the reader before the device is used again.

BSAM—The CNTRL macro should be issued after every input request.

Printer: Provides line spacing or a skip to a specific carriage control channel. A CNTRL macro cannot be used if carriage control characters are provided in the record. If the printer contains the universal character set feature, data checks should be blocked (OPTCD=U should not appear in the data control block).

Magnetic Tape: Provides method of forward spacing and backspacing (BSAM only for a data set that is not open for output). If **OPTCD=H** is indicated in the data control block, the CNTRL macro can be used to perform record positioning on DOS tapes that contain embedded checkpoint records. Embedded checkpoint records encountered during the record positioning are bypassed and are not counted as blocks spaced over. OPTCD=H must be specified in a job control language DD statement. The CNTRL macro cannot be used to backspace DOS 7-track tapes that are written in data convert mode that contain embedded checkpoint records (BSAM).

Note: The CNTRL macro should not be used with output operations on BSAM tape data sets.

3525 Printing: Provides line spacing or a skip to a specific printing line on the card. The card contains 25 printing lines; the odd-numbered lines 1 through 23 correspond to the printer skip channels 1 through 12 (see the **SK** operand). For additional information about 3525 printing operations, see OS and OS/VS Programming Support for the IBM 3505 Card Reader and IBM 3525 Card Punch.

The CNTRL macro is written:

[symbol]	CNTRL	dcb address
		{,SS,{1|2}}
		{,SP,{1|2|3}}
		{,SK,{1|2|...|11|12}}
		{,BSM}
		{,FSM}
		{,BSR[,number of blocks]}
		{,FSR[,number of blocks]}

dcb address
The dcb address operand specifies the address of the data control block for the data set opened for the on-line device.

SS,{1|2}
The **SS** operand is coded as shown to indicate that the control function requested is stacker selection on a card reader; either 1 or 2 must be coded to indicate which stacker is to be selected.

SP,{1|2|3}
The **SP** operand is coded as shown to indicate that the control function requested is printer line spacing or 3525 card punch line spacing; either 1, 2, or 3 must be coded to indicate the number of spaces for each print line.

SK,{1|2|...|11|12}
The SK operand is coded as shown to indicate that the control function requested is a skip operation on the printer or 3525 card punch, print feature; a number (1 through 12) must be coded to indicate the channel or print line to which the skip is to be taken.

BSM
The **BSM** operand indicates that the control function requested is to backspace the magnetic tape past a tapemark, then forward space over the tapemark. When this operand is specified, the DCBBLKCT field in the data control block is set to zero.

FSM
The **FSM** operand indicates that the control function requested is to forward space the magnetic tape over a tapemark, then backspace past the tapemark. When this operand is specified, the DCBBLKCT field in the data control block is set to zero.

BSR
The **BSR** operand indicates that the control function requested is to backspace the magnetic tape the number of blocks indicated in the number-of-blocks operand.

FSR
The **FSR** operand indicates that the control function requested is to forward space the magnetic tape the number of blocks indicated in the number-of-blocks operand.

number of blocks—symbol, decimal digit, absexp, or (2-12)
The number-of-blocks operand specifies the number of blocks to backspace (see **BSR** operand) or forward space (see **FSR** operand) the magnetic tape. The maximum value that can be specified is 32767. If the number-of-blocks operand is omitted, 1 is assumed.

If the forward space or backspace operation is not completed successfully, control is passed to the error analysis (SYNAD) routine; if no SYNAD exit routine is designated, the task is abnormally terminated. Register contents, when control is passed to the error analysis routine, are shown in Appendix A, "Status Information Following an Input/Output Operation" on page 192. If a tapemark is encountered for **BSR** or **FSR**, control is returned to the processing program, and register 15 contains a count of the uncompleted forward spaces or backspaces. If the operation is completed normally, register 15 contains the value zero.

DCB—CONSTRUCT A DATA CONTROL BLOCK (BDAM)

The data control block for a basic direct access method (BDAM) data set is constructed during assembly of the problem program. The **DSORG** and **MACRF** operands must be coded in the DCB macro instruction, but the other operands can be supplied from the DD statement or an existing data set label (DSCB). If more than one of these sources specifies information for a particular field, the order of priority is the DCB macro instruction, DD statement, and data set label. Each of the BDAM DCB operand descriptions contains a heading, "Source." The information under this heading describes the sources from which an operand can be supplied to the data control block.

The DCB macro for BDAM is written:

```
[symbol] | DCB | [BFALN={F|D}]
         |     | [,BFTEK=R]
         |     | [,BLKSIZE=absexp]
         |     | [,BUFCB=relexp]
         |     | [,BUFL=absexp]
         |     | [,BUFNO=absexp]
         |     | [,DDNAME=symbol]¹
         |     | ,DSORG={DA|DAU}
         |     | [,EXLST=relexp]
         |     | [,KEYLEN=absexp]
         |     | [,LIMCT=absexp]
         |     | ,MACRF={{(R{K[I]|I}[X][S][C])}
         |     |        {(W{A[K][I]|K[I]|I}[C])}
         |     |        {(R{K[I]|I}[X][S][C],W{A[K][I]|K[I]|I}[C])}
         |     | [,OPTCD={[R|A][E][F][W]]}
         |     | [,RECFM={U|V[S|BS]|F[T]}]
         |     | [,SYNAD=relexp]
```

¹ This parameter must be supplied before an OPEN macro is issued for this DCB; it cannot be supplied in the open exit routine.

The following describes the DCB operands that can be specified for creating and processing a BDAM data set:

BFALN={F|D}

The **BFALN** operand specifies the boundary alignment for each buffer in the buffer pool. The **BFALN** operand can be specified when (1) BSAM is being used to create a BDAM data set and buffers are acquired automatically, (2) when an existing BDAM data set is being processed and dynamic buffering is requested, or (3) when the GETPOOL macro instruction is used to construct the buffer pool. If the **BFALN** operand is omitted, the system provides doubleword alignment for each buffer. The following describes the characters that can be specified:

F

specifies that each buffer is aligned on a fullword boundary that is not also a doubleword boundary.

D

specifies that each buffer is aligned on a doubleword boundary.

If the BUILD macro instruction is used to construct the buffer pool or if the problem program controls all buffering, the problem program must provide the area for the buffers and control buffer alignment.

Source: The **BFALN** operand can be supplied in the DCB macro instruction, in the DCB subparameter of a DD statement, or by the problem program before completion of the data control block exit routine. If both the **BFALN** and **BFTEK**

operands are specified, they must be supplied from the same source.

BFTEK=R

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

When variable-length spanned records are written, the data length can exceed the total capacity of a single track on the direct access device being used, or it can exceed the remaining capacity on a given track. The system divides the data block into segments (if necessary), writes the first segment on a track, and writes the remaining segment(s) on the following track(s).

When a variable-length spanned record is read, the system reads each segment and assembles a complete data block in the buffer designated in the area address operand of a READ macro instruction.

Note: Variable-length spanned records can also be read using BSAM. When BSAM is used to read a BDAM variable-length spanned record, the record is read one segment at a time, and the problem program must assemble the segments into a complete data block. This operation is described in the section for the BSAM DCB macro instruction.

Source: The **BFTEK** operand can be supplied in the DCB macro instruction, in the DCB subparameter of a DD statement, or by the problem program before completion of the data control block exit routine. If both the **BFTEK** and **BFALN** operands are specified, they must be supplied from the same source.

BLKSIZE=<u>absexp</u> (maximum value is 32760)

The **BLKSIZE** operand specifies the length, in bytes, of each data block for fixed-length records, or it specifies the maximum length, in bytes, of each data block for variable-length or undefined-length records. If keys are used, the length of the key is not included in the value specified for the **BLKSIZE** operand.

The actual value that can be specified in the **BLKSIZE** operand depends on the record format and the type of direct access device being used. If track overflow is used or if variable-length spanned records are used, the value specified in the **BLKSIZE** operand can be up to the maximum. For all other record formats (F, V, VBS, and U), the maximum value that can be specified in the **BLKSIZE** operand is determined by the track capacity of a single track on the direct access device being used. Device capacity for direct access devices is described in Appendix C, "Device Capacities" on page 194 of this publication. For additional information about device capacity and space allocation, see <u>Data Administration Guide</u>.

Source: The **BLKSIZE** operand can be supplied in the DCB macro instruction, in the DCB subparameter of a DD statement, by the problem program before completion of the data control block exit routine, or by the data set label of an existing data set.

BUFCB=<u>relexp</u>

The **BUFCB** operand specifies the address of the buffer pool control block when the buffer pool is constructed by a BUILD macro instruction.

If the buffer pool is constructed automatically, dynamically, or by a GETPOOL macro instruction, the system places the address of the buffer pool control block into the data control block, and the **BUFCB** operand is not required. The **BUFCB** operand is not required if the problem program controls all buffering.

Source: The **BUFCB** operand can be supplied in the DCB macro instruction or by the problem program before completion of the data control block exit routine.

BUFL=absexp (maximum value KEYLEN + BLKSIZE is 32760)
The **BUFL** operand specifies the length, in bytes, of each buffer in the buffer pool when the buffers are acquired automatically (create BDAM) or dynamically (existing BDAM).

When buffers are acquired automatically (create BDAM), the BUFL operand is optional; if specified, the value must be at least as large as the sum of the values specified for the **KEYLEN** and **BLKSIZE** operands. If the **BUFL** operand is omitted, the system constructs buffers with a length equal to the sum of the values specified in the **KEYLEN** and **BLKSIZE** operands.

The **BUFL** operand must be specified when an existing BDAM data set is being processed and dynamic buffering is requested. Its value must be at least as large as the value specified for the **BLKSIZE** operand when the READ or WRITE macro instruction specifies a key address, or the value specified in the **BUFL** operand must be at least as large as the sum of the values specified in the **KEYLEN** and **BLKSIZE** operands if the READ and WRITE macro instructions specify **'S'** for the key address.

The **BUFL** operand can be omitted if the buffer pool is constructed by a BUILD or GETPOOL macro instruction or if the problem program controls all buffering.

Source: The **BUFL** operand can be supplied in the DCB macro instruction, in the DCB subparameter of a DD statement, or by the problem program before completion of the data control block exit routine.

BUFNO=absexp (maximum value is 255)
The **BUFNO** operand specifies the number of buffers to be constructed by a BUILD macro instruction, or it specifies the number of buffers and/or segment work areas to be acquired by the system.

If the buffer pool is constructed by a BUILD macro instruction or if buffers are acquired automatically when BSAM is used to create a BDAM data set, the number of buffers must be specified in the BUFNO operand.

If dynamic buffering is requested when an existing BDAM data set is being processed, the BUFNO operand is optional; if omitted, the system acquires two buffers.

If variable-length spanned records are being processed and dynamic buffering is requested, the system also acquires a segment work area for each buffer. If dynamic buffering is not requested, the system acquires the number of segment work areas specified in the **BUFNO** operand. If the **BUFNO** operand is omitted when variable-length spanned records are being processed and dynamic buffering is not requested, the system acquires two segment work areas.

If the buffer pool is constructed by a GETPOOL macro instruction or if the problem program controls all buffering, the BUFNO operand can be omitted, unless it is required to acquire additional segment work areas for variable-length spanned records.

Source: The **BUFNO** operand can be supplied in the DCB macro instruction, in the DCB subparameter of a DD statement, or by the problem program before completion of the data control block exit routine.

DDNAME=symbol
The **DDNAME** operand specifies the name used to identify the job control language data definition (DD) statement that defines the data set being created or processed.

Source: The **DDNAME** operand can be supplied in the DCB macro instruction or can be moved into the DCB by the problem program before an OPEN macro instruction is issued to open the data set.

DSORG={DA|DAU}
The **DSORG** operand specifies the data set organization and whether the data set contains any location-dependent information that would make it unmovable. For example, if actual device addresses are used to process a BDAM data set, the data set may be unmovable. The following describes the characters that can be specified:

DA
specifies a direct organization data set.

DAU
specifies a direct organization data set that contains location-dependent information.

When a BDAM data set is created, the basic sequential access method (BSAM) is used. The **DSORG** operand in the DCB macro instruction must be coded as **DSORG=PS** or **PSU** when the data set is created, and the DCB subparameter in the corresponding DD statement must be coded as **DSORG=DA** or **DAU**. This creates a data set with a data set label identifying it as a BDAM data set.

Source: The **DSORG** operand must be specified in the DCB macro instruction. See the preceding comment about creating a BDAM data set.

EXLST=relexp
The **EXLST** operand specifies the address of the problem program exit list. The **EXLST** operand must be specified if the problem program processes user labels during the open or close routine, if the data control block exit routine is used for additional processing, or if the DCB abend exit is used for abend condition analysis.

For the format and requirements of exit list processing, see Appendix D, "DCB Exit List Format and Contents" on page 197. For additional information about exit list processing, see Data Administration Guide.

Source: The **EXLST** operand can be supplied in the DCB macro instruction or by the problem program before the exit is needed.

KEYLEN=absexp (maximum value is 255)
The **KEYLEN** operand specifies the length, in bytes, of all keys used in the data set. When keys are used, a key is associated with each data block in the data set. If the key length is not supplied by any source, no input or output requests that require a key can be specified in a READ or WRITE macro instruction.

Source: The **KEYLEN** operand can be supplied in the DCB macro instruction, in the DCB subparameter of a DD statement, by the problem program before the completion of the data control block exit routine, or by an existing data set label. If **KEYLEN=0** is specified in the DCB macro instruction, a special indicator is set in **RECFM** so that **KEYLEN** cannot be supplied from the DCB subparameter of a DD statement or data set label of an existing data set. **KEYLEN=0** can be coded only in the DCB macro instruction and will be ignored if specified in the DD statement.

LIMCT=absexp

The **LIMCT** operand specifies the number of blocks or tracks to be searched when the extended search option (**OPTCD=E**) is requested.

When the extended search option is requested and relative block addressing is used, the records must be fixed-length record format. The system converts the number of blocks specified in the LIMCT operand into the number of tracks required to contain the blocks, then proceeds in the manner described below for relative track addressing.

When the extended search option is requested and relative track addressing is used (or the number of blocks has been converted to the number of tracks), the system searches for two things: (a) the block specified in a READ or WRITE macro instruction (type **DK**), or (b) available space where it can add a block (WRITE macro instruction, type **DA**). The search is as follows:

1. The search begins at the track specified by the block address operand of a READ or WRITE macro instruction.

2. The search continues until the search is satisfied, the number of tracks specified in the **LIMCT** operand have been searched, or the entire data set has been searched. If the search has not been satisfied when the last track of the data set is reached, the system continues the search by starting at the first track of the data set if the EOF marker is on the last track that was allocated to the data set. (This operation allows the number specified in the **LIMCT** operand to exceed the size of the data set, causing the entire data set to be searched.) You can ensure that the EOF marker is on the last allocated track by determining the size of the data set and allocating space in blocks, or by allocating space in tracks and including the RLSE parameter on the SPACE operand of the DD statement (RLSE specifies that all unused tracks be returned to the system).

The problem program can change the DCBLIMCT field in the data control block at any time, but, if the extended search option is used, the DCBLIMCT field must not be zero when a READ or WRITE macro instruction is issued.

If the extended search option is not requested, the system ignores the **LIMCT** operand, and the search for a data block is limited to a single track.

Source: The **LIMCT** operand can be supplied in the DCB macro instruction, the DCB subparameter of a DD statement, or by the problem program before the count is required by a READ or WRITE macro instruction.

MACRF={{(R{K[I]|I}[X][S][C])}
{(W{A[K][I]|K[I]|I}[C])}
{(R{K[I]|I}[X][S][C],W{A[K][I]|K[I]|I}[C])}}

The **MACRF** operand specifies the type of macro instructions (READ, WRITE, CHECK, and WAIT) used when the data set is processed. The **MACRF** operand also specifies the type of search argument and BDAM functions used with the data set. When BSAM is used to create a BDAM data set, the BSAM operand **MACRF=WL** is specified. This special operand invokes the BSAM routine that can create a BDAM data set. The following describes the characters that can be specified for BDAM:

A

specifies that data blocks are to be added to the data set.

C

specifies that CHECK macro instructions are used to test for completion of read and write operations. If **C** is not specified, WAIT macro instructions must be used to test for completion of read and write operations.

I

specifies that the search argument is to be the block identification portion of the data block. If relative addressing is used, the system converts the relative address to a full device address (MBBCCHHR) before the search.

K

specifies that the search argument is to be the key portion of the data block. The location of the key to be used as a search argument is specified in a READ or WRITE macro instruction.

R

specifies that READ macro instructions are to be used. READ macro instructions can be issued when the data set is opened for INPUT, OUTPUT, or UPDAT.

S

specifies that dynamic buffering is requested by specifying **'S'** in the area address operand of a READ or WRITE macro instruction.

W

specifies that WRITE macro instructions are used. WRITE macro instructions can be issued only when the data set is opened for OUTPUT or UPDAT.

X

specifies that READ macro instructions request exclusive control of a data block. When exclusive control is requested, the data block must be released by a subsequent WRITE or RELEX macro instruction.

Source: The **MACRF** operand must be supplied in the DCB macro instruction.

OPTCD={[R|A][E][F][W]}

The **OPTCD** operand specifies the optional services that are to be used with the BDAM data set. These options are related to the type of addressing used, the extended search option, block position feedback, and write-validity checking. The following describes the characters that can be specified (the characters can be specified in any order, and no commas are allowed between characters):

A

specifies that actual device addresses (MBBCCHHR) are provided to the system when READ or WRITE macro instructions are issued.

E

specifies that the extended search option is used to locate data blocks or available space where a data block can be added. When the extended search option is specified, the number of blocks or tracks to be searched must be specified in the **LIMCT** operand. The extended search option is ignored if actual addressing (**OPTCD=A**) is also specified. The extended search option requires that the data set have keys and that the search be made by key (by specifying **DK** in the READ or WRITE macro or **DA** in the WRITE macro).

F

specifies that block position feedback requested by a READ or WRITE macro instruction is to be in the same form that was originally presented to the system in the READ or WRITE macro instruction. If the **F** operand is omitted, the system provides feedback, when requested, in the form of an 8-byte actual device address. (Feedback is always provided if exclusive control is requested.)

R

specifies that relative block addresses (in the form of 3-byte binary numbers) are provided to the system when a READ or WRITE macro instruction is issued.

W

specifies that the system is to perform a validity check for each record written.

Note: Relative track addressing can only be specified by omitting both **A** and **R** from the OPTCD operand. If you want to specify relative track addressing after your data set has been accessed using another addressing scheme (**OPTCD=A** or **R**), you should either specify a valid OPTCD operand (**E**, **F**, or **W**) in the DCB macro or DD card when you reopen your data set, or zero out the **OPTCD=A** or **R** bits in the data control block exit routine. Note that the first method will prevent the open routines from merging any of the other OPTCD bits from the format-1 DSCB in the DCB. Both methods will update the OPTCD in the DSCB if the open is for OUTPUT, OUTIN, or UPDAT.

Source: The **OPTCD** operand can be supplied in the DCB macro instruction, in the DCB subparameter of a DD statement, or by the problem program before completion of the DCB open exit routine.

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

The **RECFM** operand specifies the format and characteristics of the records in the data set. The following describes the characters that can be coded (if the optional characters are coded, they must be coded in the order shown above):

B

specifies that the data set contains blocked records. The record format **RECFM=VBS** is the only combination in which **B** can be specified. **RECFM=VBS** does not cause the system to process spanned records; the problem program must block and segment the records. **RECFM=VBS** is treated as a variable-length record by BDAM.

F

specifies that the data set contains fixed-length records.

S

specifies that the data set contains variable-length spanned records when it is coded as **RECFM=VS**. When **RECFM=VBS** is coded, the records are treated as variable-length records, and the problem program must block and segment the records.

T

specifies that track overflow is to be used with the data set. Track overflow allows a record to be partially written on one track and the remainder is written on the following track (if required).

U

specifies that the data set contains undefined-length records.

V

specifies that the data set contains variable-length records.

Source: The **RECFM** operand can be supplied in the DCB macro instruction, in the DCB subparameter of a DD statement, the problem program before completion of the data control block exit routine, or by the data set label of an existing data set.

SYNAD=relexp

The **SYNAD** operand specifies the address of the error analysis routine to be given control when an uncorrectable input/output error occurs. The contents of the registers when the error analysis routine is given control are described in Appendix A, "Status Information Following an Input/Output Operation" on page 192.

The error analysis routine must not use the save area pointed to by register 13, because this area is used by the system. The system does not restore registers when it regains control from the error analysis routine. The error analysis routine can issue a RETURN macro instruction that uses the address in register 14 to return control to the system. When control is returned in this manner, the system returns control to the problem program and proceeds as though no error had been encountered. When a BDAM data set is being created, a return from the error analysis routine to the system causes abnormal termination of the task.

If the **SYNAD** operand is omitted, the task is abnormally terminated when an uncorrectable input/output error occurs.

Source: The **SYNAD** operand can be supplied in the DCB macro instruction or by the problem program. The problem program can also change the error routine address at any time.

DCB—CONSTRUCT A DATA CONTROL BLOCK (BISAM)

The data control block for the basic indexed sequential access method (BISAM) is constructed during assembly of the problem program. The **DSORG** and **MACRF** operands must be coded in the DCB macro instruction, but the other DCB operands can be supplied from other sources. Each BISAM DCB operand description contains a heading, "Source." The information under this heading describes the sources of the operand that can be supplied to the data control block.

The DCB macro for BISAM is written:

[symbol]	**DCB**	**[BFALN={F|D}]**
		[,BUFCB=relexp**]**
		[,BUFL=absexp**]**
		[,BUFNO=absexp**]**
		[,DDNAME=symbol**]**[1]
		,DSORG=IS
		[,EXLST=relexp**]**
		,MACRF={{(R[S][C])}
		{(W{U[A]|A}[C])}
		{(R[U[S]|S][C],W{U[A]|A}[C])}}
		[,MSHI=relexp**]**
		[,MSWA=relexp**]**
		[,NCP=absexp**]**
		,OPTCD={([L][R][W])}
		[,SMSI=absexp**]**
		[,SMSW=absexp**]**
		[,SYNAD=relexp**]**

1 This parameter must be supplied before an OPEN macro is issued for this DCB; it cannot be supplied in the open exit routine.

The following describes the DCB operands that can be supplied when the basic indexed sequential access method is used:

BFALN={F|D}

The **BFALN** operand specifies the boundary alignment for each buffer in the buffer pool when the buffer pool is acquired for use with dynamic buffering or when the buffer pool is constructed by a GETPOOL macro instruction. If the **BFALN** operand is omitted, the system provides doubleword alignment for each buffer. The following describes the characters that can be specified:

F

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

D

specifies that each buffer is on a doubleword boundary.

If the BUILD macro instruction is used to construct the buffer pool, or the problem program controls all buffering, the problem program must provide an area for the buffers and control buffer alignment.

Source: The **BFALN** operand can be supplied in the DCB macro instruction, in the DCB subparameter of a DD statement, or by the problem program before completion of the data control block exit routine.

BUFCB=relexp

The **BUFCB** operand specifies the address of the buffer pool control block when the buffer pool is constructed by a BUILD macro instruction.

If dynamic buffering is requested or the buffer pool is constructed by a GETPOOL macro instruction, the system places the address of the buffer pool control block into the data control block, and the **BUFCB** operand must be omitted. The **BUFCB** operand must be omitted if the problem program controls all buffering.

Source: The **BUFCB** operand can be supplied in the DCB macro instruction or by the problem program before completion of the data control block exit routine.

BUFL=<u>absexp</u> (maximum value is 32760)

The **BUFL** operand specifies the length of each buffer to be constructed by a BUILD or GETPOOL macro instruction. When the data set is opened, the system computes the minimum length required and verifies that the length in the buffer pool control block is equal to or greater than the minimum required. The system then inserts the computed length into the **BUFL** field of the data control block.

If dynamic buffering is requested, the system computes the buffer length required, and the **BUFL** operand is not required.

If the problem program controls all buffering, the **BUFL** operand is not required. However, an ISAM data set requires additional buffer space for system use. For a description of the buffer length required for various ISAM operations, see <u>Data Administration Guide</u>.

Source: The **BUFL** operand can be supplied in the DCB macro instruction, in the DCB subparameter of a DD statement, or by the problem program before completion of the data control block exit routine.

BUFNO=<u>absexp</u> (maximum value is 255)

The **BUFNO** operand specifies the number of buffers requested for use with dynamic buffering, or it specifies the number of buffers to be constructed by a BUILD macro instruction. If dynamic buffering is requested but the **BUFNO** operand is omitted, the system automatically acquires two buffers for use with dynamic buffering.

If the GETPOOL macro instruction is used to construct the buffer pool, the **BUFNO** operand is not required.

Source: The **BUFNO** operand can be supplied in the DCB macro instruction, in the DCB subparameter of a DD statement, or by the problem program before completion of the data control block exit routine.

DDNAME=<u>symbol</u>

The **DDNAME** operand specifies the name used to identify the job control language data definition statement that defines the ISAM data set to be processed.

Source: The **DDNAME** operand can be supplied in the DCB macro instruction or by the problem program before an OPEN macro instruction is issued to open the data set.

DSORG=IS

The **DSORG** operand specifies the indexed sequential organization of the data set. **IS** is the only combination of characters that can be coded for BISAM.

Source: Unless it is for a data set passed from a previous job step, the **DSORG** operand must be coded in the DCB macro instruction and in the DCB subparameter of a DD statement. In this case, **DSORG** may be omitted from the DD statement.

EXLST=<u>relexp</u>

The **EXLST** operand specifies the address of the problem program exit list. If the problem program uses the data control block exit routine for additional processing, the **EXLST** operand is required.

For the format and requirements for exit list processing, see Appendix D, "DCB Exit List Format and Contents" on page 197. For additional information about exit list processing, see Data Administration Guide.

Source: The **EXLST** operand can be supplied in the DCB macro instruction or by the problem program before the associated exit is required.

MACRF={{(R[S][C])}
{(W{U[A]|A}[C])}
{(R[U[S]|S][C],W{U[A]|A}[C])}}

The MACRF operand specifies the type of macro instructions (READ, WRITE, CHECK, WAIT, and FREEDBUF) and type of processing (add records, dynamic buffering, and update records) to be used with the data set being processed. The operand can be coded in any of the combinations shown above; the following describes the characters that can be coded:

A
specifies that new records are to be added to the data set. This character must be coded if WRITE KN macro instructions are used with the data set.

C
specifies that the CHECK macro instruction is used to test I/O operations for completion. If **C** is not coded, WAIT macro instructions must be used.

R
specifies that READ macro instructions are to be used.

S
specifies that dynamic buffering is requested in READ macro instructions. **S** should not be specified if the problem program provides the buffer pool.

U
specifies that records in the data set will be updated in place. If **U** is coded in combination with R, it must also be coded in combination with **W.** For example, **MACRF=(RU,WU)**.

W
specifies that WRITE macro instructions are to be used.

Source: The **MACRF** operand must be coded in the DCB macro instruction.

MSHI=relexp
The MSHI operand specifies the address of the storage area used to contain the highest-level master index for the data set. The system uses this area to reduce the search time required to find a given record in the data set. The **MSHI** operand is coded only when the **SMSI** operand is coded.

Source: The **MSHI** operand can be supplied in the DCB macro instruction or by the problem program before completion of the data control block exit routine.

MSWA=relexp
The MSWA operand specifies the address of the storage work area to be used by the system when new records are being added to the data set. This operand is optional, but the system acquires a minimum-size work area if the operand is omitted. The MSWA operand is coded only when the SMSW operand is coded.

Processing efficiency can be increased if more than a minimum-size work area is provided. For more detailed information about work area size, see Data Administration Guide.

Note: QISAM uses the DCBMSWA, DCBSMSI, and DCBSMSW fields in the data control block as a work area; these fields contain meaningful information only when the data set is opened for BISAM.

Source: The MSWA operand can be supplied in the DCB macro instruction or by the problem program before completion of the data control block exit routine.

NCP=absexp (maximum value is 99)

The **NCP** operand specifies the maximum number of READ/WRITE macro instructions that are issued before the first CHECK (or WAIT) macro instruction is issued to test for completion of the I/O operation. The maximum number that can be specified may be less than 99, depending on the amount of virtual storage available in the region. If the **NCP** operand is omitted, 1 is assumed. If dynamic buffering is used, the value specified for the **NCP** operand must not exceed the number of buffers specified in the **BUFNO** operand.

Source: The **NCP** operand can be supplied in the DCB macro instruction, in the DCB subparameter of a DD statement, or by the problem program before completion of the data control block open exit routine.

OPTCD={[L][R][W]}

The **OPTCD** operand specifies the optional services performed by the control program. All optional services must be requested by one method; that is, by the data set label of an existing data set, this macro, or the DD statement on the DCB parameter. However, it can be modified by the problem program. The characters may be coded in any order and, when used in combination, no commas are permitted between characters.

L

specifies that the control program delete records that have a first byte of all 1's. (These records can be deleted when space is required for new records. To use the delete option, the relative key position (RKP) must be greater than 0 for fixed-length records and greater than 4 for variable-length records.)

R

specifies that the control program place reorganization criteria information in certain fields of the data control block. (The problem program can analyze these statistics to determine when to reorganize the data set. This option is provided whenever the **OPTCD** operand is omitted from all sources.)

W

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

SMSI=absexp (maximum value is 65535)

The **SMSI** operand specifies the length, in bytes, required to contain the highest-level master index for the data set being processed. The size required can be determined from the DCBNCRHI field of the data control block. When an ISAM data set is created (with QISAM), the size of the highest-level index is inserted into the DCBNCRHI field. If the value specified in the **SMSI** operand is less than the value in the DCBNCRHI field, the task is abnormally terminated.

Note: QISAM uses the DCBMSWA, DCBSMSI, and DCBSMSW fields as a work area; these fields contain meaningful information only when the data set is opened for BISAM.

Source: The **SMSI** operand can be supplied in the DCB macro instruction or by the problem program before completion of the data control block exit routine.

SMSW=<u>absexp</u> (maximum value is 65535)

The **SMSW** operand specifies the length, in bytes, of a work area that is used by BISAM. This operand is optional, but the system acquires a minimum-size work area if the operand is omitted. The **SMSW** operand is coded only when the MSWA operand is also coded. If the **SMSW** operand is coded but the size specified is less than the minimum required, the task is abnormally terminated. <u>Data Administration Guide</u> describes the methods of calculating the size of the work area.

If unblocked records are used, the work area must be large enough to contain all the count fields (8 bytes each), key fields, and data fields contained on one direct access device track.

If blocked records are used, the work area must be large enough to contain all the count fields (8 bytes each) and data fields contained on one direct access device track plus additional space for one logical record (**LRECL** value).

Note: QISAM uses the DCBMSWA, DCBSMSI, and DCBSMSW fields in the data control block as a work area; these fields contain meaningful information only when the data set is opened for BISAM.

Source: The **SMSW** operand can be supplied in the DCB macro instruction or by the problem program before completion of the data control block exit routine.

SYNAD=<u>relexp</u>

The **SYNAD** operand specifies the address of the error analysis routine given control when an uncorrectable input/output error occurs. The contents of the registers when the error analysis routine is given control are described in Appendix A, "Status Information Following an Input/Output Operation" on page 192.

The error analysis routine must not use the save area pointed to by register 13 because this area is used by the system. The system does not restore registers when it regains control from the error analysis routine. The error analysis routine can issue a RETURN macro instruction that uses the address in register 14 to return control to the system. When control is returned in this manner, the system returns control to the problem program and proceeds as though no error had been encountered. If the error analysis routine continues processing, the results are unpredictable.

If the **SYNAD** operand is omitted, the task is abnormally terminated when an uncorrectable input/output error occurs.

Source: The **SYNAD** operand can be supplied in the DCB macro instruction or by the problem program. The problem program can also change the error analysis routine address at any time.

DCB—CONSTRUCT A DATA CONTROL BLOCK (BPAM)

The data control block for the basic partitioned access method (BPAM) is constructed during assembly of the problem program. The **DSORG** and **MACRF** operands must be specified in the DCB macro instruction, but the other DCB operands can be supplied from other sources. Each of the BPAM DCB operand descriptions contains a heading, "Source." The information under this heading describes the sources that can supply the operand to the data control block.

The DCB macro for BPAM is written:

[symbol]	**DCB**	**[BFALN={F|D}]** **[,BLKSIZE=**absexp**]** **[,BUFCB=**relexp**]** **[,BUFL=**absexp**]** **[,BUFNO=**absexp**]** **[,DDNAME=**symbol**]**[1] **,DSORG={PO|POU}** **[,EODAD=**relexp**]** **[,EXLST=**relexp**]** **[,KEYLEN=**absexp**]** **[,LRECL=**absexp**]** **,MACRF={(R|W|R,W)}**[1] **[,NCP=**absexp**]** **[,OPTCD={{C|W[C]}** **{C|H[C]}** **{C|W[H][C]}}]** **[,RECFM={{U[T][A|M]}** **{V[B[T]|T][A|M]}** **{F[B[T]|T][A|M]}}]** **[,SYNAD=**relexp**]**

[1] This parameter must be supplied before an OPEN macro is issued for this DCB; it cannot be supplied in the open exit routine.

When you create or process a BPAM data set, you can specify the following DCB operands:

BFALN={F|D}

The **BFALN** operand specifies the boundary alignment for each buffer in the buffer pool when the buffer pool is constructed automatically or by a GETPOOL macro instruction. If the **BFALN** operand is omitted, the system provides doubleword alignment for each buffer. The characters that can be specified in the **BFALN** operand are:

F

specifies that each buffer is aligned on a fullword boundary that is not also a doubleword boundary.

D

specifies that each buffer is aligned on a doubleword boundary.

If the BUILD macro instruction is used to construct the buffer pool or if the problem program controls all buffering, the problem program must provide an area for the buffers and control buffer alignment.

Source: The **BFALN** operand can be supplied in the DCB macro instruction, in the DCB subparameter of a DD statement, or by the problem program before completion of the data control block exit routine.

BLKSIZE=absexp (maximum value KEYLEN + BLKSIZE is 32760)

The **BLKSIZE** operand specifies the length, in bytes, of each data block for fixed-length records, or it specifies the maximum length, in bytes, for variable-length or undefined-length records. If keys are used, the length of the key is not included in the value specified for the **BLKSIZE** operand.

The actual block size that can be specified depends on the record format and the type of direct access device being used. If track overflow is used, the block size can be up to the maximum. If track overflow is not used, the maximum block size is determined by the track capacity of a single track on the direct access device being used. Device capacity for direct access devices is described in Appendix C, "Device Capacities" on page 194. For additional information about device capacity and space allocation, see Data Administration Guide.

For variable-length records, the value specified in the **BLKSIZE** operand must include the maximum logical record length (up to 32756 bytes) plus 4 bytes for the block descriptor word (BDW).

For undefined-length records, the value specified for the **BLKSIZE** operand can be altered by the problem program when the actual length becomes known to the problem program. The value can be inserted into the DCBBLKSI field of the data control block or specified in the length operand of a READ/WRITE macro instruction.

Source: The **BLKSIZE** operand can be supplied in the DCB macro instruction, in the DCB subparameter of a DD statement, by the problem program before completion of the data control block exit routine, or by the data set label of an existing data set.

BUFCB=relexp
The **BUFCB** operand specifies the address of the buffer pool control block when the buffer pool is constructed by a BUILD macro instruction.

If the buffer pool is constructed automatically or by a GETPOOL macro instruction, the system places the address of the buffer pool control block into the data control block and the **BUFCB** operand can be omitted. Also, if the problem program controls all buffering, the **BUFCB** operand should be omitted.

Source: The **BUFCB** operand can be supplied in the DCB macro instruction or by the problem program before completion of the data control block exit routine.

BUFL=absexp (maximum value is 32760)
The **BUFL** operand specifies the length, in bytes, of each buffer in the buffer pool when the buffer pool is acquired automatically. If the **BUFL** operand is omitted and the buffer pool is acquired automatically, the system acquires buffers with a length that is equal to the sum of the values specified in the **KEYLEN** and **BLKSIZE** operands. If the problem program requires longer buffers, the **BUFL** operand should be specified.

If the problem program controls all buffering, the **BUFL** operand is not required.

Source: The **BUFL** operand can be supplied in the DCB macro instruction, in the DCB subparameter of a DD statement, or by the problem program before completion of the data control block exit routine.

BUFNO=absexp (maximum value is 255)
The **BUFNO** operand specifies the number of buffers to be constructed by a BUILD macro instruction, or it specifies the number of buffers to be acquired automatically by the system.

If the problem program controls all buffering or if the buffer pool is constructed by a GETPOOL macro instruction, the **BUFNO** operand should be omitted.

Source: The **BUFNO** operand can be supplied in the DCB macro instruction, in the DCB subparameter of a DD statement, or by the problem program before completion of the data control block exit routine.

DDNAME=symbol

The **DDNAME** operand specifies the name used to identify the job control language data definition (DD) statement that defines the data set being created or processed.

Source: The **DDNAME** operand can be supplied in the DCB macro instruction or by the problem program before an OPEN macro instruction is issued to open the data set.

DSORG={PO|POU}

The **DSORG** operand specifies the data set organization and whether the data set contains any location-dependent information that would make it unmovable. The characters that can be specified are:

PO

specifies a partitioned data set organization.

POU

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

Note: If BSAM or QSAM is used to add or retrieve a single member of a partitioned data set, a sequential access method is being used, and the **DSORG** operand is specified as **PS** or **PSU.** The name of the member being processed in this manner is supplied in a DD statement.

Source: The **DSORG** operand must be specified in the DCB macro instruction.

EODAD=relexp

The **EODAD** operand specifies the address of the routine given control when the end of the input data set is reached. Control is given to this routine when an input request is made (READ macro instruction) and there are no additional input records to retrieve. The routine is entered when a CHECK macro instruction is issued and the end of the data set is reached. If the end of the data set is reached and no **EODAD** address has been supplied, the task is abnormally terminated. For additional information on the EODAD routine, see Data Administration Guide.

Source: The **EODAD** operand can be supplied in the DCB macro instruction or by the problem program before the end of the data set is reached.

EXLST=relexp

The **EXLST** operand specifies the address of the problem program exit list. The **EXLST** operand is required if the problem program uses the data control block exit routine for additional processing or if the DCB abend exit is used for abend condition analysis.

For the format and requirements of the exit list processing, see Appendix D, "DCB Exit List Format and Contents" on page 197. For additional information about exit list processing, see Data Administration Guide.

Source: The **EXLST** operand can be supplied in the DCB macro instruction or by the problem program before the OPEN macro instruction is issued to open the data set.

KEYLEN=<u>absexp</u> (maximum value is 255)
The **KEYLEN** operand specifies the length, in bytes, of the key associated with each data block in the direct access device data set. If the key length is not supplied from any source by the end of the data control block exit routine, a key length of zero (no keys) is assumed.

Source: The **KEYLEN** operand can be supplied in the DCB macro instruction, in the DCB subparameter of a DD statement, by the problem program before the completion of the data control block exit routine, or by the data set label of an existing data set. If **KEYLEN=0** is specified in the DCB macro instruction, a special indicator is set in **RECFM** so that **KEYLEN** cannot be supplied from the DCB subparameter of a DD statement or data set label of an existing data set. **KEYLEN=0** can be coded only in the DCB macro instruction and will be ignored if specified in the DD statement.

LRECL=<u>absexp</u> (maximum value is 32760)
The **LRECL** operand specifies the length, in bytes, of each fixed-length logical record in the data set; It is required only for fixed-length records. The value specified in the **LRECL** operand cannot exceed the value specified in the **BLKSIZE** operand.

If the records are unblocked, the value specified in the **LRECL** operand must equal the value specified in the **BLKSIZE** operand. If the records are blocked, the value specified in the **LRECL** operand must be evenly divisible into the value specified in the **BLKSIZE** operand.

Source: The **LRECL** operand can be supplied in the DCB macro instruction, in the DCB subparameter of a DD statement, by the problem program before completion of the data control block exit routine, or by the data set label of an existing data set.

MACRF={(R|W|R,W)}
The **MACRF** operand specifies the type of macro instructions (READ, WRITE, and NOTE/POINT) that are used to process the data set. The characters that can be specified are:

R
specifies that READ macro instructions are to be used. This operand automatically provides the capability to use both the NOTE and POINT macro instructions with the data set.

W
specifies that WRITE macro instructions are to be used. This operand automatically provides the capability to use both the NOTE and POINT macro instructions with the data set.

All BPAM READ and WRITE macro instructions issued must be tested for completion using a CHECK macro instruction. The **MACRF** operand does not require any coding to specify that a CHECK macro instruction will be used.

Source: The **MACRF** operand must be specified in the DCB macro instruction.

NCP=<u>absexp</u> (maximum value is 99)
The **NCP** operand specifies the maximum number of READ and WRITE macro instructions that will be issued before the first CHECK macro instruction is issued. The maximum number may be less than 99, depending on the amount of virtual storage available in the region. If chained scheduling is specified, the value of **NCP** determines the maximum number of channel program segments that can be chained and must be specified as more than 1. If the **NCP** operand is omitted, 1 is assumed.

Source: The **NCP** operand can be supplied in the DCB macro instruction, in the DCB subparameter of a DD statement, or by the problem program before completion of the data control block open exit routine.

OPTCD={{C|W[C]}
{C|H[C]}
{C|W[H][C]}}

The **OPTCD** operand specifies the optional services performed by the system. The characters that can be specified (in any order, in any combination, and without commas between characters) are:

C
: specifies that chained scheduling is used. This option is ignored for direct access devices.

H
: If OPTCD=H is coded in the DCB parameters of a DD statement, H specifies that, if a partitioned data set is being opened for input and resides on an MSS device, then, at OPEN time, the data set is to be staged to EOF on the virtual DASD device.

W
: specifies that the system is to perform a validity check for each record written.

Source: The **OPTCD** operand can be supplied in the DCB macro instruction, in the DCB subparameter of a DD statement, or by the problem program before an OPEN macro instruction is issued to open the data set. However, all optional services must be requested from the same source.

RECFM={{U[T][A|M]}
{V[B[T]]|[T][A|M]}
{F[B[T]]|[T][A|M]}}

The **RECFM** operand specifies the record format and characteristics of the data set being created or processed. All the record formats shown above can be specified, but in those formats that show blocked records, the problem program must perform the blocking and deblocking of logical records; BPAM recognizes only data blocks. The characters that can be specified are:

A
: specifies that the records in the data set contain ISO/ANSI/FIPS control characters. For a description of control characters, see Appendix E, "Control Characters" on page 199.

B
: specifies that the data set contains blocked records.

F
: specifies that the data set contains fixed-length records.

M
: specifies that the records in the data set contain machine code control characters. For a description of control characters, see Appendix E, "Control Characters" on page 199.

T
: specifies that track overflow is used with the data set. Track overflow allows a record to be written partially on one track of a direct access device and the remainder of the record written on the following track (if required). Chained scheduling (**OPTCD=C**) cannot be used if the track overflow is used.

U
: specifies that the data set contains undefined-length records.

V
: specifies that the data set contains variable-length records.

Source: The **RECFM** operand can be supplied in the DCB macro instruction, in the DCB subparameter of a DD statement, by the problem program before completion of the data control block exit routine, or by the data set label of an existing data set.

SYNAD=relexp
: The **SYNAD** operand specifies the address of the error analysis (SYNAD) routine to be given control when an uncorrectable input/output error occurs. The contents of the registers when the error analysis routine is given control are described in Appendix A, "Status Information Following an Input/Output Operation" on page 192.

The error analysis routine must not use the save area pointed to by register 13, because this area is used by the system. The system does not restore registers when it regains control from the error analysis routine. The error analysis routine can return control to the system by issuing a RETURN macro instruction. If control is returned to the system, the system returns control to the problem program and proceeds as though no error had been encountered.

If the **SYNAD** operand is omitted, the task is abnormally terminated when an uncorrectable input/output error occurs.

Source: The **SYNAD** operand can be supplied in the DCB macro instruction or by the problem program. The problem program can also change the error routine address at any time.

DCB—CONSTRUCT A DATA CONTROL BLOCK (BSAM)

The data control block for the basic sequential access method (BSAM) is constructed during assembly of the problem program. The **DSORG** and **MACRF** operands must be coded in the DCB macro instruction, but the other DCB operands can be supplied to the data control block from other sources. Each DCB operand description contains a heading, "Source." The information under this heading describes the sources of an operand that can be supplied.

The DCB macro for BSAM is written:

```
[symbol]   DCB        [BFALN={F|D}]
                      [,BFTEK=R]
                      [,BLKSIZE=absexp]
                      [,BUFCB=relexp]
                      [,BUFL=absexp]
                      [,BUFNO=absexp]
                      [,BUFOFF={absexp|L}]
                      [,DDNAME=symbol]¹
                      [,DEVD={{DA
                               [,KEYLEN=absexp]}
                             {TA
                               [,DEN={1|2|3|4}]
                               [,TRTCH={C|E|ET|T}]}
                             {PR
                               [,PRTSP={0|1|2|3}]}
                             {PC
                               [,MODE=[C|E]]
                               [,STACK={1|2}]
                               [,FUNC={I|P|PW[XT]|R|RP[D]|
                                         RW[T]|RWP[XT][D]|W[T]}]
                             {RD
                               [,MODE=[C|E][O|R]]
                               [,STACK={1|2}]
                               [,FUNC={I|P|PW[XT]|R|RP[D]|
                                         RW[T]|RWP[XT][D]|W[T]}]}]
                      ,DSORG={PS|PSU}¹
                      [,EODAD=relexp]
                      [,EXLST=relexp]
                      [,KEYLEN=absexp]
                      [,LRECL={absexp|X}]
                      ,MACRF={{(R[C|P])}
                             {(W[C|P|L])}
                             {(R[C|P],W[C|P])}}¹
                      [,NCP=absexp]
                      [,OPTCD={{B}
                               {T}
                               {U[C]}
                               {C[T][B][U]}
                               {H[Z][B]}
                               {J[C][U]}
                               {W[C][T][B][U]}
                               {Z[C][T][B][U]}
                               {Q[C][B][T|Z]}}]
                      [,RECFM={{U[T][A|M]}
                               {V[B|S|T|BS|BT][A|M]}
                               {D[B|S|BS][A]}
                               {F[B|S|T|BS|BT][A|M]}}]
                      [,SYNAD=relexp]
```

¹ This parameter must be supplied before an OPEN macro is issued for this DCB; it cannot be supplied in the open exit routine.

The following describes the operands that can be specified in the DCB macro instruction for a BSAM data set:

BFALN={F|D}
The **BFALN** operand specifies the boundary alignment for each buffer in the buffer pool when the buffer pool is constructed automatically or by a **GETPOOL** macro instruction. If the **BFALN** operand is omitted, the system provides doubleword alignment for each buffer.

If the data set being created or processed contains ISCII/ASCII tape records with a block prefix, the block prefix is entered at the beginning of the buffer, and data alignment depends on the length of the block prefix. For a description of how to specify the block prefix length, see the description of the DCB **BUFOFF** operand.

The characters that can be specified are:

F
specifies that each buffer is on a fullword boundary that is not also a doubleword boundary.

D
specifies that each buffer is on a doubleword boundary.

If the BUILD macro instruction is used to construct the buffer pool or if the problem program controls all buffering, the problem program must provide an area for the buffers and control buffer alignment.

Source: The **BFALN** operand can be supplied in the DCB macro instruction, in the DCB subparameter of a DD statement, or by the problem program before completion of the data control block exit routine. If both the **BFALN** and **BFTEK** operands are specified, they must be supplied by the same source.

BFTEK=R
The **BFTEK=R** operand specifies that BSAM is used to read unblocked variable-length spanned records with keys from a BDAM data set. Each read operation reads one segment of the record and places it in the area designated in the READ macro instruction. The first segment enters at the beginning of the area, but all subsequent segments are offset by the length of the key (only the first segment has a key). The problem program must provide an area in which it can assemble a record, identify each segment, and assemble the segments into a complete record.

Source: The **BFTEK** operand can be supplied in the DCB macro instruction, in the DCB subparameter of a DD statement, or by the problem program before completion of the data control block exit routine. If both the **BFTEK** and **BFALN** operands are specified, they must be supplied from the same source.

BLKSIZE=absexp (maximum value KEYLEN + BLKSIZE is 32760)
The **BLKSIZE** operand specifies the maximum block length in bytes. For fixed-length, unblocked records, this operand specifies the record length. The **BLKSIZE** operand includes only the data block length; if keys are used, the length of the key is not included in the value specified for the **BLKSIZE** operand.

The actual value that can be specified in the **BLKSIZE** operand depends on the device type and the record format being used. Device capacity is shown in Appendix C, "Device Capacities" on page 194. For additional information about device capacity, see Data Administration Guide.

For direct access devices when track overflow is used or variable-length spanned records are being processed, the value specified in the **BLKSIZE** operand can be up to the maximum value. For other record formats used with direct access devices, the value specified for **BLKSIZE** cannot exceed the capacity of a single track.

If fixed-length records are used, the value specified in the **BLKSIZE** operand should be an integral multiple of the value specified for the logical record length (LRECL).

If variable-length records are used, the value specified in the **BLKSIZE** operand must include the maximum logical record length (up to 32756 bytes) plus the 4 bytes required for the block descriptor word (BDW). For format-D variable-length records (ISCII/ASCII data sets), the minimum value for **BLKSIZE** is 18 bytes. The maximum value is 2048 bytes. For additional information about the **BLKSIZE** restrictions, see Data Administration Guide.

If ISCII/ASCII tape records with a block prefix are processed, the value specified in the **BLKSIZE** operand must also include the length of the block prefix.

If BSAM is used to read variable-length spanned records from a BDAM data set, the value specified for the **BLKSIZE** operand must be as large as the longest possible record segment in the BDAM data set, including 4 bytes for the segment descriptor word (SDW) and 4 bytes for the block descriptor word (BDW).

If undefined-length records are used, the value specified for the **BLKSIZE** operand can be altered by the problem program when the actual length becomes known to the problem program. The value can be inserted directly into the DCBBLKSI field of the data control block or specified in the length operand of a READ/WRITE macro instruction.

Source: The **BLKSIZE** operand can be supplied in the DCB macro instruction, in the DCB subparameter of a DD statement, by the problem program before completion of the data control block exit routine, or by the data set label of an existing data set.

Note: The maximum block size for Version 3 ISO/ANSI/FIPS tapes (ISO 1001-1979 and ANSI X3.27-1978) is 2048 bytes. An attempt to exceed 2048 bytes for a Version 3 tape results in a label validation installation exit being taken.

BUFCB=relexp

The **BUFCB** operand specifies the address of the buffer pool control block in a buffer pool constructed by a BUILD macro instruction.

If the buffer pool is to be constructed automatically or by a GETPOOL macro instruction, the system places the address of the buffer pool control block into the data control block, and the **BUFCB** operand should be omitted. If the problem program is to control all buffering, the **BUFCB** operand is not required.

Source: The **BUFCB** operand can be supplied in the DCB macro instruction or by the problem program before completion of the data control block exit routine.

BUFL=absexp (maximum value is 32760)

The **BUFL** operand specifies the length, in bytes, for each buffer in the buffer pool when the buffer pool is acquired automatically. The system acquires buffers with a length equal to the sum of the values specified in the **KEYLEN** and **BLKSIZE** operands if the **BUFL** operand is omitted; if the problem program requires larger buffers, the **BUFL** operand must be specified. If the **BUFL** operand is specified, it must be at least as large as the value specified in the **BLKSIZE** operand. If the data set is for card image mode, the **BUFL** operand should be specified as 160. The description of the **DEVD** operand contains a description of card image mode.

If the data set contains ISCII/ASCII tape records with a block prefix, the value specified in the **BUFL** operand must include the block length plus the length of the block prefix.

If the problem program is to control all buffering or if the buffer pool is to be constructed by a GETPOOL or BUILD macro instruction, the **BUFL** operand is not required.

Source: The **BUFL** operand can be supplied in the DCB macro instruction, in the **DCB** keyword on a DD statement, or by the problem program before completion of the data control block exit routine.

BUFNO=absexp (maximum value is 255)
The **BUFNO** operand specifies the number of buffers constructed by a BUILD macro instruction or the number of buffers to be acquired automatically by the system.

If the problem program controls all buffering or if the buffer pool is constructed by a GETPOOL macro instruction, the **BUFNO** operand should be omitted.

Source: The **BUFNO** operand can be supplied in the DCB macro instruction, in the DCB subparameter of a DD statement, or by the problem program before completion of the data control block exit routine.

BUFOFF={absexp|**L}**
The **BUFOFF** operand specifies the length, in bytes, of the block prefix used with an ISCII/ASCII tape data set. When BSAM is used to read an ISCII/ASCII tape data set, the problem program must use the block prefix length to determine the location of the data in the buffer. When BSAM is used to write an output ISCII/ASCII tape data set, the problem program must insert the block prefix into the buffer, followed by the data (BSAM considers the block prefix as data). The block prefix and data can consist of any characters that can be translated into 7-bit ISCII/ASCII code; any character that cannot be translated is replaced with a substitute character. (For a more detailed description of ISCII/ASCII translation characteristics, see Magnetic Tape Labels and File Structure.) For format-D records, the RDW must be binary; if RECFM=D and BUFOFF=L, the RDW and BDW must both be binary. On output, the control program translates the BDW and RDW to ISCII/ASCII characters and, on input, the control program converts ISCII/ASCII data to BDW and RDW. The following can be specified in the **BUFOFF** operand:

absexp
specifies the length, in bytes, of the block prefix. This value can be from 0 to 99 for an input data set. The value must be 0 for writing an output data set with fixed-length or undefined-length records (BSAM considers the block prefix part of the data record).

L
specifies that the block prefix is 4 bytes long and contains the block length. **BUFOFF=L** is used when format-D records (ISCII/ASCII) are processed. When **BUFOFF=L** is specified, the BSAM problem program can process the data records (using READ and WRITE macro instructions) in the same manner as if the data were in format-V variable-length records. For further information on this operand, see "Variable-Length Records—Format D" in Data Administration Guide.

If the **BUFOFF** operand is omitted for an input data set with format-D records, the system inserts the record length into the DCBLRECL field of the data control block; the problem program must obtain the length from this field to process the record.

If the **BUFOFF** operand is omitted from an output data set with format-D records, the problem program must insert the actual record length into the DCBBLKSI field of the data control block or specify the record length in the length operand of a WRITE macro instruction.

Source: The **BUFOFF** operand can be supplied in the DCB macro instruction, in the DCB subparameter of a DD statement, or by the problem program before an OPEN macro instruction is issued to open the data set. BUFOFF=absexp can also be supplied by the label of an existing data set; BUFOFF=L cannot be supplied by the label of an existing data set.

DDNAME=symbol
The **DDNAME** operand specifies the name used to identify the job control language data definition (DD) statement that defines the data set being created or processed.

Source: The **DDNAME** operand can be supplied in the DCB macro instruction or by the problem program before an OPEN macro instruction is issued to open the data set.

DEVD={DA|TA|PR|PC|RD}[,options]
The **DEVD** operand specifies the device type where the data set can or does reside. The device types above are shown with the optional operand(s) that can be coded when a particular device is used. The devices are listed in order of device independence. For example, if **DEVD=DA** is coded in a DCB macro instruction (or the **DEVD** operand is omitted, which causes a default to **DA**), the data control block constructed during assembly could later be used for any of the other devices, but, if **DEVD=RD** is coded, the data control block can be used only with a card reader or card reader punch. Unless you are certain that device interchangeability is not required, you should either code **DEVD=DA** or omit the operand and allow it to default to **DA**.

If system input is directed to an intermediate storage device, the **DEVD** operand is omitted, and the job control language for the problem program designates the system input device to be used. Likewise, if system output is directed to an intermediate storage device, the **DEVD** operand is omitted, and the job control language for the problem program designates the system output device to be used. If DEVD=PR, PC, or RD is coded, the DCB macro should not be coded within the first 16 bytes of addressability for the control section.

The **DEVD** operand is discussed below according to individual device type:

DEVD=DA
[,KEYLEN=absexp]
specifies that the data control block can be used for a direct access device (or any of the other device types described following DA).

KEYLEN=absexp
The **KEYLEN** operand can be specified only for data sets that reside on direct access devices. Because the KEYLEN is usually coded without a **DEVD** operand (default taken), the description of the **KEYLEN** operand is in alphabetic sequence with the other operands.

DEVD=TA
[,DEN={1|2|3|4}]
[,TRTCH={C|E|ET|T}]
Specifies that the data control block can be used for a magnetic tape data set (or any of the other device types described following TA). If **TA** is coded, the following optional operands can be coded:

DEN={1|2|3|4}
The **DEN** operand specifies the recording density in the number of bits-per-inch per track as shown in the following:

Recording Density

DEN	**7-Track**	**9-Track**	**18-Track**
1	556	N/A	N/A
2	800	800 (NRZI)[1]	N/A
3	N/A	1600 (PE)[2]	N/A
4	N/A	6250 (GCR)[3]	N/A

[1] NRZI is for nonreturn-to-zero inverted mode.

[2] PE is for phase encoded mode.

[3] GCR is for group coded recording mode.

If the **DEN** operand is not supplied by any source, the highest applicable density is assumed.

TRTCH={C|E|ET|T}
The **TRTCH** operand specifies the recording technique for 7-track tape. One of the above 4-character combinations can be coded. If the **TRTCH** operand is omitted, odd parity with no translation or conversion is assumed. The characters that can be specified are:

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

E
specifies even parity with no translation or conversion.

ET
specifies even parity with BCDIC to EBCDIC translation required and no data-conversion feature.

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

DEVD=PR
[,PRTSP={0|1|2|3}]
Specifies that the data control block is used for an on-line printer (or any of the other device types following PR). If **PR** is coded, the following optional operand can be coded:

PRTSP={0|1|2|3}
The **PRTSP** operand specifies the line spacing on the printer. This operand is not valid if the **RECFM** operand specifies either machine (**RECFM=M**), ISO/ANSI/FIPS (**RECFM=A**) control characters. If the **PRTSP** operand is not specified from any source, 1 is assumed.

The characters that can be specified are:

0
: specifies that spacing is suppressed (no space).

1
: specifies single spacing.

2
: specifies double spacing (one blank line between printed lines).

3
: specifies triple spacing (two blank lines between printed lines).

DEVD=PC
[,MODE=[C|E]
[,STACK={1|2}]
[,FUNC={I|P|PW[XT]|R|RP[D]|RW[T]|RWP[XT][D]|W[T]}]

Specifies that the data control block is used for a card punch (or any of the other device types following **PC**). If **PC** is coded, the following optional operands can be specified:

MODE=[C|E]

The **MODE** operand specifies the mode of operation for the card punch. The characters that can be specified (if the **MODE** operand is omitted, **E** is assumed) are:

C
: specifies that the cards are to be punched in card image mode. In card image mode, the 12 rows in each card column are punched from two consecutive bytes in virtual storage. Rows 12 through 3 are punched from the low-order 6 bits of one byte and rows 4 through 9 are punched from the low-order 6 bits of the following byte.

E
: specifies that cards are to be punched in EBCDIC code.

STACK={1|2}

The **STACK** operand specifies the stacker bin where the card is placed after punching is completed. If this operand is omitted, stacker number 1 is used. The characters that can be specified are:

1
: specifies stacker number 1.

2
: specifies stacker number 2.

FUNC={I|P|PW[XT]|R|RP[D]|RW[T]|RWP[XT][D]|W[T]}

The **FUNC** operand defines the type of 3525 card punch data sets that are used. If the **FUNC** operand is omitted from all sources, a data set opened for input defaults to read only, and a data set opened for output defaults to punch only. The characters that can be specified in the **FUNC** operand are:

D
: specifies that the data protection option is to be used. The data protection option prevents punching information into card columns that already contain data. When the data protection option is used, an 80-byte data protection image (DPI) must have been

previously stored in SYS1.IMAGELIB. Data protection applies only to the output/punch portion of a read and punch or read punch and print operation.

I
: specifies that the data in the data set is to be punched into cards and printed on the cards; the first 64 characters are printed on line 1 of the card and the remaining 16 characters are printed on line 3.

P
: specifies that the data set is for punching cards. See the description of the character X for associated punch and print data sets.

R
: specifies that the data set is for reading cards.

T
: specifies that the two-line print option is used. The two-line print option allows two lines of data to be printed on the card (lines 1 and 3). If **T** is not specified, the multiline print option is used; this allows printing on all 25 possible print lines. In either case, the data printed may be the same as the data punched in the card, or it may be entirely different data.

W
: specifies that the data set is for printing. See the description of the character X for associated punch and print data sets.

X
: specifies that an associated data set is opened for output for both punching and printing. Coding the character X is used to distinguish the 3525 printer output data set from the 3525 punch output data set.

Note: If data protection is specified, the data protection image (DPI) must be specified in the FCB parameter of the DD statement for the data set.

DEVD=RD
[,MODE=[C|E][O|R]]
[,STACK={1|2}]
[,FUNC={I|P|PW[XT]|R|RP[D]|RW[T]|RWP[XT][D]|W[T]}]

Specifies that the data control block is used with a card reader or card read punch. If **RD** is specified, the data control block cannot be used with any other device type. When **RD** is coded, the following optional operands can be specified:

MODE=[C|<u>E</u>][O|R]

The **MODE** operand specifies the mode of operation for the card reader. The characters that can be specified are:

C
: specifies that the cards to be read are in card image mode. In card image mode, the 12 rows in each card column are read into two consecutive bytes of virtual storage. Rows 12 through 3 are read into one byte and rows 4 through 9 are read into the following byte.

E
: specifies that the cards to be read contain data in EBCDIC code.

O
: specifies that the program runs in optical-mark-read mode (3505 card reader).

R
: specifies that the program runs in read-column-eliminate mode (3505 card reader or 3525 card punch, read feature).

Note: If the **MODE** operand for a 3505 or 3525 is specified in the DCB subparameter of a DD statement, either **C** or **E** must be specified if **R** or **O** is specified.

STACK={1|2}
The **STACK** operand specifies the stacker bin where the card is placed after reading is completed. If this operand is omitted, stacker number 1 is used. The characters that can be specified are:

1
: specifies stacker number 1.

2
: specifies stacker number 2.

FUNC={I|P|PW[XT]|R|RP[D]|RW[T]|RWP[XT][D]|W[T]}
The **FUNC** operand defines the type of 3525 card punch data sets that are used. If the **FUNC** operand is omitted from all sources, a data set opened for input defaults to read only, and a data set opened for output defaults to punch only. The characters that can be specified in the **FUNC** operand are:

D
: specifies that the data protection option is to be used. The data protection option prevents punching information into card columns that already contain data. When the data protection option is used, an 80-byte data protection image (DPI) must have been previously stored in SYS1.IMAGELIB. Data protection applies only to the output/punch portion of a read and punch or read punch and print operation.

I
: specifies that the data in the data set is to be punched into cards and printed on the cards; the first 64 characters are printed on line 1 of the card and the remaining 16 characters are printed on line 3.

P
: specifies that the data set is for punching cards. See the description of the character X for associated punch and print data sets.

R
: specifies that the data set is for reading cards.

T
: specifies that the two-line print option is used. The two-line print option allows two lines of data to be printed on the card (lines 1 and 3). If **T** is not specified, the multiline print option is used; this allows printing on all 25 possible print lines. In

either case, the data printed may be the same as the data punched in the card, or it may be entirely different data.

W

specifies that the data set is for printing. See the description of the character X for associated punch and print data sets.

X

specifies that an associated data set is opened for output for both punching and printing. Coding the character X is used to distinguish the 3525 printer output data set from the 3525 punch output data set.

Note: If data protection is specified, the data protection image (DPI) must be specified in the FCB subparameter of the DD statement for the data set.

Source: The **DEVD** operand can be supplied only in the DCB macro instruction. However, the optional operands can be supplied in the DCB macro instruction, the DCB subparameter of a DD statement, or by the problem program before completion of the data control block exit routine.

DSORG={PS|PSU}

The **DSORG** operand specifies the organization of the data set and if the data set contains any location-dependent information that would make it unmovable. The following can be specified:

PS

specifies a physical sequential data set.

PSU

specifies a physical sequential data set that contains location-dependent information that would make it unmovable.

Source: The **DSORG** operand must be coded in the DCB macro instruction.

EODAD=relexp

The **EODAD** operand specifies the address of the routine given control when the end of an input data set is reached. If the record format is **RECFM=FS** or **FBS**, the end-of-data condition is sensed when a file mark is read or when more data is requested after reading a truncated block. The end-of-data routine is entered when the CHECK macro instruction determines that the READ macro instruction reached the end of the data. If the end of the data set is reached but no EODAD address has been supplied, the task is abnormally terminated. For additional information on the EODAD routine, see Data Administration Guide.

When the data set has been opened for UPDAT and volumes are to be switched, the problem program should issue a FEOV macro instruction after the EODAD routine has been entered.

Source: The **EODAD** operand can be supplied in the DCB macro instruction or by the problem program before the end of the data set is reached.

EXLST=relexp

The **EXLST** operand specifies the address of the problem program exit list. The **EXLST** operand is required if the problem program requires additional processing for user labels, user totaling, data control block exit routine, end-of-volume, block count exits, to define a forms control buffer (FCB) image, use the JFCBE exit (for the IBM 3800

Printing Subsystem), or to use the DCB abend exit for abend condition analysis.

For the format and requirements of exit list processing, see Appendix D, "DCB Exit List Format and Contents" on page 197. For additional information about exit list processing, see Data Administration Guide.

Source: The **EXLST** operand can be supplied in the DCB macro instruction or by the problem program any time before the exit is required by the problem program.

KEYLEN=absexp (maximum value is 255)
The **KEYLEN** operand specifies the length, in bytes, for the key associated with each data block in a direct access device data set. If the key length is not supplied from any source before completion of the data control block exit routine, a key length of zero (no keys) is assumed.

Source: The **KEYLEN** operand can be supplied in the DCB macro instruction, in the DCB subparameter of a DD statement, by the problem program before the completion of the data control block exit routine, or by the data set label of an existing data set. If **KEYLEN=0** is specified in the DCB macro instruction, a special indicator is set in **RECFM** so that **KEYLEN** cannot be supplied from the DCB subparameter of a DD statement or data set label of an existing data set. **KEYLEN=0** can be coded only in the DCB macro instruction and will be ignored if specified in the DD statement.

LRECL={absexp**|X}**
The **LRECL** operand specifies the length, in bytes, for fixed-length records, or it specifies the maximum length, in bytes, for variable-length records. **LRECL=X** is used for variable-length spanned records that exceed 32756 bytes. Except when variable-length spanned records are used, the value specified for the **LRECL** operand cannot exceed the value specified for the **BLKSIZE** operand.

Except when variable-length spanned records are used, the **LRECL** operand can be omitted for BSAM; the system uses the value specified in the **BLKSIZE** operand. If the **LRECL** value is coded, it is coded as described in the following.

For fixed-length records that are unblocked, the value specified in the **LRECL** operand should be equal to the value specified in the **BLKSIZE** operand. For blocked fixed-length records, the value specified in the **LRECL** operand should be evenly divisible into the value specified in the **BLKSIZE** operand. However, the **LRECL** operand will not be validity checked.

For variable-length records, the value specified in **LRECL** must include the maximum data length (up to 32752 bytes) plus 4 bytes for the RDW.

For undefined-length records, the **LRECL** operand should be omitted; the actual length can be supplied dynamically in a READ/WRITE macro instruction. When an undefined-length record is read, the actual length of the record is returned by the system in the DCBLRECL field of the data control block.

When BSAM is used to create a BDAM data set with variable-length spanned records, the **LRECL** value should be the maximum data length (up to 32752) plus four bytes for the record descriptor word (RDW), or, if the logical record length is greater than 32756 bytes, **LRECL=X** is specified.

Source: The **LRECL** operand can be supplied in the DCB macro instruction, in the DCB subparameter of a DD statement, by the problem program before completion of the data control block exit routine, or by the data set label of an existing data set.

```
MACRF={{(R[C|P])}
       {(W[C|P|L])}
       {(R[C|P],W[C|P])}}
```

The MACRF operand specifies the type of macro instructions (READ, WRITE, CNTRL, and NOTE/POINT) that are used with the data set being created or processed. The BSAM MACRF operand also provides the special form (MACRF=WL) for creating a BDAM data set. The MACRF operand can be coded in any of the forms shown above. The following characters can be coded:

C

specifies that the CNTRL macro instruction is used with the data set. If **C** is specified to be used with a card reader, a CNTRL macro instruction must follow every input request.

L

specifies that BSAM is used to create a BDAM data set. This character can be specified only in the combination **MACRF=WL**.

P

specifies that POINT macro instructions are used with the data set being created or processed. Specifying **P** in the MACRF operand also automatically provides the capability of using NOTE macro instructions with the data set. **P** should not be coded for SYSIN or SYSOUT data sets. (See explanations of the NOTE and POINT macro instructions.)

R

specifies that READ macro instructions are to be used.

W

specifies that WRITE macro instructions are to be used.

Note: Each READ and WRITE macro instruction issued in the problem program must be checked for completion by a CHECK macro instruction.

Source: The MACRF operand must be specified in the DCB macro instruction.

NCP=absexp (maximum value is 99)

The **NCP** operand specifies the maximum number of READ/WRITE macro instructions that will be issued before the first CHECK macro instruction is issued to test for completion of the I/O operation. The maximum number may be less than 99, depending on the amount of virtual storage available in the region. If the **NCP** operand is omitted, 1 is assumed.

Source: The **NCP** operand can be supplied in the DCB macro instruction, in the DCB subparameter of a DD statement, or by the problem program before completion of the data control block open exit routine.

```
OPTCD={{B}
      {T}
      {U[C]}
      {C[T][B][U]}
      {H[Z][B]}
      {J[C][U]}
      {W[C][T][B][U]}
      {Z[C][T][B][U]}
      {Q[C][B][T]}
      {Z}}
```

The **OPTCD** operand specifies the optional services that are used with the BSAM data set. Two of the optional services, **OPTCD=B** and **OPTCD=H**, cannot be specified in the DCB macro instruction. They are requested in the DCB subparameter of a DD statement. Because all optional services requests must be supplied by the same source, the **OPTCD** operand must be omitted from the DCB macro instruction if either of these options is requested in a DD statement. The characters that can be specified (in any order, in one of the combinations shown above, and without commas between characters) are:

C

requests that chained scheduling be used. **OPTCD=C** cannot be specified if **BFTEK=R** is specified for the same data control block. Also, chained scheduling cannot be specified for associated data sets or printing on a 3525 and is ignored for direct access devices.

Note: Except where it is not allowed, chained scheduling is used whether requested or not. For conditions under which chained scheduling is not allowed, see Data Administration Guide.

J

specifies that the first data byte in the output data line will be a 3800 table reference character. This table reference character selects a particular character arrangement table for the printing of the data line and can be used singly or in conjunction with ISO, ANSI, or machine control characters. This option is valid only for the IBM 3800 Printing Subsystem. For information on the table reference character and character arrangement table modules, see IBM 3800 Printing Subsystem Programmer's Guide.

Q

requests that ISCII/ASCII tape records in an input data set be converted to EBCDIC code after the input record has been read. Translation is done at CHECK time for input. It also requests that an output record in EBCDIC code be converted to ISCII/ASCII code before the record is written. For further information on this conversion, see "Variable-Length Records—Format D" in Data Administration Guide.

The Q option is unconditionally set by open routines if the data set is for a tape with ISO/ANSI/FIPS labels. For more information about ISCII/ASCII to EBCDIC or EBCDIC to ISCII/ASCII translations, see Magnetic Tape Labels and File Structure Administration.

T

requests the user totaling function. If this function is requested, the **EXLST** operand should specify the address of an exit list to be used. **T** cannot be specified for SYSIN and SYSOUT data sets.

U

is specified only for a printer with the universal character set (UCS) feature or the 3800 Printing Subsystem. This option unblocks data checks (permits them to be recognized as errors) and allows analysis by the appropriate error analysis routine (SYNAD exit routine). If the **U** option is omitted, data checks are not recognized as errors.

For the IBM Mass Storage System (MSS): U requests window processing to reduce the amount of staging space required to process large sequential data sets on MSS. DSORG must specify physical sequential, allocation must be in cylinders, and type of I/O accessing must be either INPUT only or OUTPUT only.

W

for DASD, specifies that the system is to perform a validity check on each record written on a direct access device. For buffered devices, specifies that device end interrupt is to be given only when a record is physically on the device. By specifying OPTCD=W with buffered devices, you do not benefit from the performance advantage of buffering.

Z

requests, for magnetic tape, input only, the system to shorten its normal error recovery procedure to consider a data check as a permanent I/O error after five unsuccessful attempts to read a record. This option is available only if it has also been specified as a SYSGEN option. **OPTCD=Z** is meant to be used when a tape is known to contain errors and there is no need to process every record. The error analysis routine (SYNAD) should keep a count of permanent errors and terminate processing if the number becomes excessive.

Note: The following describes the optional services that can be requested in the DCB subparameter of a DD statement. If either of these options is requested, the complete **OPTCD** operand must be supplied in the DD statement.

B

If **OPTCD=B** is specified in the DCB subparameter of a DD statement, it forces the end-of-volume (EOV) routine to disregard the end-of-file recognition for magnetic tape. When this occurs, the EOV routine uses the number of volume serial numbers to determine end of file.

H

If **OPTCD=H** is specified in the DCB subparameter of a DD statement, it specifies that the DOS/OS interchange feature is being used with the data set.

Source: The **OPTCD** operand can be supplied in the DCB macro instruction, in the DCB subparameter of a DD statement, in the data set label for direct access devices, or by the problem program before completion of the DCB open exit routine or JFCBE exit routine. However, all optional services must be requested from the same source.

RECFM={{U[T][A|M]}
{V[B|S|T|BS|BT][A|M]}
{D[B|S|BS][A]}
{F[B|S|T|BS|BT][A|M]}}

The **RECFM** operand specifies the record format and characteristics of the data set being created or processed. All the record formats shown above can be specified, but in those record formats that specify blocked records, the problem program must perform the blocking and deblocking of logical records; BSAM recognizes only data blocks. The following describes the characters that can be specified:

A

specifies that the records in the data set contain International Organization for Standardization (ISO) or American National Standards Institute (ANSI) control characters. For a description of control characters, see Appendix E, "Control Characters" on page 199.

B

specifies that the data set contains blocked records.

D

specifies that the data set contains variable-length ISCII/ASCII tape records.

F
specifies that the data set contains fixed-length records.

M
specifies that the records in the data set contain machine code control characters. For a description of control characters, see Appendix E, "Control Characters" on page 199. **RECFM=M** cannot be used with ISCII/ASCII data sets.

S
S specifies, for fixed-length records, that the records are to be written as standard blocks; with the exception of the last block or track in the data set, the data set contains no truncated blocks or unfilled tracks. Do not code **S** to retrieve records from a data set that was created using a **RECFM** other than standard.

For variable-length records, including variable-length ISCII/ASCII, **S** specifies that a record can span more than one block.

T
specifies that track overflow is used with the data set. Track overflow allows a record to be written partially on one track of a direct access device and the remainder of the record to be written on the following track or tracks as required. Chained scheduling cannot be requested if track overflow is used.

U
specifies that the data set contains undefined-length records.

Note: Format-U records are not supported for Version 3 ISO/ANSI/FIPS tapes. An attempt to process a format-U record for a Version 3 tape results in a label validation installation exit being taken.

Only ISO/ANSI Version 1 (ISO 1001-1969 or ANSI X3.27-1969) format-U records can be used for input.

V
specifies that the data set contains variable-length records.

Notes:

- **RECFM=V** cannot be specified for a card reader data set or an ISO/ANSI/FIPS tape data set.
- **RECFM=VBS** does not provide the spanned record function; if this format is used, the problem program must block and segment the records.
- **RECFM=DBS** or **RECFM=DS** does not provide the spanned record function; if this format is used, the problem program must block and segment the records.
- **RECFM=VS, VBS, DS,** or **DBS** cannot be specified for a SYSIN data set.
- **RECFM=V** cannot be used for a 7-track tape unless the data conversion feature (**TRTCH=C**) is used.

Source: The **RECFM** operand can be supplied in the DCB macro instruction, in the DCB subparameter of a DD statement, by the problem program before completion of the data control block exit routine, or by the data set label of an existing data set.

SYNAD=relexp
The **SYNAD** operand specifies the address of the error analysis (SYNAD) routine to be given control if an uncorrectable input/output error occurs. The contents of the registers when the error analysis routine is given control are described in Appendix A, "Status Information Following an Input/Output Operation" on page 192.

The error analysis routine must not use the save area pointed to by register 13, because this area is used by the system. The system does not restore registers when it regains control from the error analysis routine. The error analysis routine can issue a RETURN macro instruction that uses the address in register 14 to return control to the system. If control is returned to the system, the system returns control to the problem program and proceeds as though no error had been encountered.

If the **SYNAD** operand is omitted, the task is abnormally terminated when an uncorrectable input/output error occurs.

Source: The **SYNAD** operand can be supplied in the DCB macro instruction or by the problem program. The problem program can also change the error routine address at any time.

When operating a directly allocated 3800 Model 3 using all-points addressability, the SYNAD routine will be entered if Print Services Facility (PSF) detects an unrecoverable error. However, no error information is available to the SYNAD routine for a directly allocated 3800 Model 3. If you want to continue processing, you must close and reopen the data set to restart PSF. For more information on the 3800 Model 3, see IBM 3800 Printing Subsystem Programmer's Guide for Models 3 and 8.

DCB—CONSTRUCT A DATA CONTROL BLOCK (QISAM)

The data control block for a queued indexed sequential access method (QISAM) data set is constructed during assembly of the problem program. The **DSORG** and **MACRF** operands must be coded in the DCB macro instruction, but the other DCB operands can be supplied from other sources. Each QISAM DCB operand description contains a heading, "Source." The information under this heading describes the sources that can supply the operand to the data control block.

The DCB macro for QISAM is written:

```
[symbol]   DCB        [BFALN={F|D}]
                      [,BLKSIZE=absexp]
                      [,BUFCB=relexp]
                      [,BUFL=absexp]
                      [,BUFNO=absexp]
                      [,CYLOFL=absexp]
                      [,DDNAME=symbol]¹
                      ,DSORG={IS|ISU}
                      [,EODAD=relexp]
                      [,EXLST=relexp]
                      [,KEYLEN=absexp]
                      [,LRECL=absexp]
                      ,MACRF={{(PM)}
                             {(PL)}
                             {(GM[,S{K|I}])}
                             {(GL[,S{K|I}][,PU])}}
                      [,NTM=absexp]
                      [,OPTCD=[I][L][M][R][U][W][Y]]
                      [,RECFM={V[B]|F[B]}]
                      [,RKP=absexp]
                      [,SYNAD=relexp]
```

¹ This parameter must be supplied before an OPEN macro is issued for this DCB; it cannot be supplied in the open exit routine.

The following describes the DCB operands that can be specified when a QISAM data set is being created or processed:

BFALN={F|D}

The **BFALN** operand specifies the alignment of each buffer in the buffer pool when the buffer pool is constructed automatically or by a GETPOOL macro instruction. If the **BFALN** operand is omitted, the system provides doubleword alignment for each buffer. The following describes the characters that can be specified:

F

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

D

specifies that each buffer is on a doubleword boundary.

If the **BUILD** macro instruction is used to construct the buffer pool, the problem program must provide a storage area for the buffers and control buffer alignment.

Source: The **BFALN** operand can be supplied in the DCB macro instruction, in the DCB subparameter of a DD statement, or by the problem program before completion of the data control block exit routine.

BLKSIZE=absexp (maximum value KEYLEN + BLKSIZE is 32760)

The **BLKSIZE** operand specifies the length, in bytes, for each data block when fixed-length records are used, or it specifies the maximum length in bytes, for each data block when variable-length records are used. The **BLKSIZE** operand must be specified when an ISAM data set is created. When an existing ISAM data set is processed, the **BLKSIZE** operand must be omitted (it is supplied by the data set label).

Track capacity of the direct access device being used must be considered when the block size for an ISAM data set is specified. For fixed-length records, the sum of the key length, data length, and device overhead plus 10 bytes (for ISAM use) must not exceed the capacity of a single track on the direct access device being used. For variable-length records, the sum of the key length, block-descriptor word length, record-descriptor word length, data length, and device overhead plus 10 bytes (for ISAM use) must not exceed the capacity of a single track on the direct access device being used. Device capacity and device overhead are described in Appendix C, "Device Capacities" on page 194. For additional information about device capacity and space allocation, see Data Administration Guide.

If fixed-length records are used, the value specified in the **BLKSIZE** operand must be an integral multiple of the value specified in the **LRECL** operand.

Source: When an ISAM data set is created, the **BLKSIZE** operand can be supplied in the DCB macro instruction, in the DCB subparameter of a DD statement, or by the problem program before completion of the data control block exit routine. When an existing ISAM data set is processed, the **BLKSIZE** operand must be omitted from the other sources, allowing the data set label to supply the value.

BUFCB=relexp
The **BUFCB** operand specifies the address of the buffer pool control block constructed by a BUILD macro instruction.

If the system constructs the buffer pool automatically or if the buffer pool is constructed by a GETPOOL macro instruction, the system places the address of the buffer pool control block into the data control block, and the **BUFCB** operand should be omitted.

Source: The **BUFCB** operand can be supplied in the DCB macro instruction or by the problem program before completion of the data control block exit routine.

BUFL=absexp (maximum value is 32760)
The **BUFL** operand specifies the length, in bytes, of each buffer in the buffer pool when the buffer pool is constructed by a BUILD or GETPOOL macro instruction. When the data set is opened, the system computes the minimum buffer length required and verifies that the length in the buffer pool control block is equal to or greater than the minimum length required. The system then inserts the computed length into the data control block.

The **BUFL** operand is not required for QISAM if the system acquires buffers automatically; the system computes the minimum buffer length required and inserts the value into the data control block.

If the buffer pool is constructed with a BUILD or GETPOOL macro instruction, additional space is required in each buffer for system use. For a description of the buffer length required for various ISAM operations, see Data Administration Guide.

Source: The **BUFL** operand can be supplied in the DCB macro instruction, in the DCB subparameter of a DD statement, or by the problem program before completion of the data control block exit routine.

BUFNO=absexp (maximum value is 255)
The **BUFNO** operand specifies the number of buffers to be constructed by a BUILD macro instruction, or it specifies

the number of buffers to be acquired automatically by the system. If the **BUFNO** operand is omitted, the system automatically acquires two buffers.

If the GETPOOL macro instruction is used to construct the buffer pool, the **BUFNO** operand is not required.

Source: The **BUFNO** operand can be supplied in the DCB macro instruction, in the DCB subparameter of a DD statement, or by the problem program before completion of the data control block exit routine.

CYLOFL=absexp (maximum value is number of tracks minus 1)
The **CYLOFL** operand specifies the number of tracks on each cylinder that is reserved as an overflow area. The overflow area is used to contain records that are forced off prime area tracks when additional records are added to the prime area track in ascending key sequence. ISAM maintains pointers to records in the overflow area so that the entire data set is logically in ascending key sequence. Tracks in the cylinder overflow area are used by the system only if **OPTCD=Y** is specified. For a more complete description of cylinder overflow area, refer to the space allocation section of Data Administration Guide.

Source: When an ISAM data set is created, the **CYLOFL** operand can be supplied in the DCB macro instruction, in the DCB subparameter of a DD statement, or by the problem program before completion of the data control block exit routine. When an existing ISAM data set is processed, the **CYLOFL** operand should be omitted, allowing the data set label to supply the operand.

DDNAME=symbol
The **DDNAME** operand specifies the name used to identify the job control language data definition (DD) statement that defines the data set being created or processed.

Source: The **DDNAME** operand can be supplied in the DCB macro instruction or by the problem program before an OPEN macro instruction is issued to open the data set.

DSORG={IS|ISU}
The **DSORG** operand specifies the organization of the data set and indicates if the data set contains any location-dependent information that would make it unmovable. The following characters can be specified:

IS
specifies an indexed sequential data set organization.

ISU
specifies an indexed sequential data set that contains location-dependent information. **ISU** can be specified only when an ISAM data set is created.

Source: The **DSORG** operand must be specified in the DCB macro instruction. When an ISAM data set is created, DSORG=IS or ISU must also be specified in the DCB subparameter of the corresponding DD statement.

EODAD=relexp
The **EODAD** operand specifies the address of the routine to be given control when the end of an input data set is reached. For ISAM, this operand would apply only to scan mode when a data set is open for an input operation. Control is given to this routine when a GET macro instruction is issued and there are no more input records to retrieve. For additional information on the EODAD routine, see Data Administration Guide.

Source: The **EODAD** operand can be supplied in the DCB macro instruction or by the problem program before the end of the data set is reached.

EXLST=<u>relexp</u>
The **EXLST** operand specifies the address of the problem program exit list. The **EXLST** operand is required only if the problem program uses the data control block exit routine for additional processing.

For the format and requirements for exit list processing, see Appendix D, "DCB Exit List Format and Contents" on page 197. For additional information about exit list processing, see <u>Data Administration Guide</u>.

Source: The **EXLST** operand can be supplied in the DCB macro instruction or by the problem program before the associated exit is required.

KEYLEN=<u>absexp</u> (maximum value is 255)
The **KEYLEN** operand specifies the length, in bytes, of the key associated with each record in an indexed sequential data set. When blocked records are used, the key of the last record in the block (highest key) is used to identify the block. However, each logical record within the block has its own identifying key that ISAM uses to access a given logical record.

Source: When an ISAM data set is created, the **KEYLEN** operand can be supplied in the DCB macro instruction, in the DCB subparameter of a DD statement, or by the problem program before completion of the data control block exit routine. When an existing ISAM data set is processed, the **KEYLEN** operand must be omitted, allowing the data set level to supply the key length value. **KEYLEN=0** is not valid for an ISAM data set.

LRECL=<u>absexp</u> (maximum value is device-dependent)
The **LRECL** operand specifies the length, in bytes, for fixed-length records, or it specifies the maximum length, in bytes, for variable-length records. The value specified in the **LRECL** operand cannot exceed the value specified in the **BLKSIZE** operand. When fixed, unblocked records are used and the relative key position (as specified in the **RKP** operand) is zero, the value specified in the **LRECL** operand should include only the data length (the key is not written as part of the fixed, unblocked record when **RKP=0**).

The track capacity of the direct access device being used must be considered if maximum-length logical records are being used. For fixed-length records, the sum of the key length, data length, and device overhead plus 10 bytes (for ISAM use) must not exceed the capacity of a single track on the direct access device being used. For variable-length records, the sum of the key length, data length, device overhead, block-descriptor-word length, and record-descriptor-word length plus 10 bytes (for ISAM use) must not exceed the capacity of a single track on the direct access device being used. Device capacities are shown in Appendix C, "Device Capacities" on page 194. For additional information about device capacity and space allocation, see <u>Data Administration Guide</u>.

Source: When an ISAM data set is created, the **LRECL** operand can be supplied in the DCB macro instruction, in the DCB subparameter of a DD statement, or by the problem program before completion of the data control block exit routine. When an existing ISAM data set is processed, the **LRECL** operand must be omitted, allowing the data set label to supply the value.

```
MACRF={{(PM)}
       {(PL)}
       {(GM[,S{K|I}])}
       {(GL[,S{K|I}][,PU])}}
```
The **MACRF** operand specifies the type of macro instructions, the transmittal mode, and type of search to be used with the data set being processed. The operand can be coded in

any of the combinations shown above; the following describes the characters that can be coded.

The following characters can be specified only when the data set is being created (load mode) or additional records are being added to the end of the data set (resume load):

PL
: specifies that PUT macro instructions are used in the locate transmittal mode; the system provides the problem program with the address of a buffer containing the data to be written into the data set.

PM
: specifies that PUT macro instructions are used in the move transmittal mode; the system moves the data to be written from the problem program work area to the buffer being used.

The following characters can be specified only when the data set is being processed (scan mode) or when records in an ISAM data set are being updated in place:

GL
: specifies that GET macro instructions are used in the locate transmittal mode; the system provides the problem program with the address of a buffer containing the logical record read.

GM
: specifies that GET macro instructions are used in the move mode; the system moves the logical record from the buffer to the problem program work area.

I
: specifies that actual device addresses (MBBCCHHR) are used to search for a record (or the first record) to be read.

K
: specifies that a key or key class is used to search for a record (or the first record) to be read.

PU
: specifies that PUTX macro instructions are to be used to return updated records to the data set.

S
: specifies that SETL macro instructions are used to set the beginning location for processing the data set.

Source: The **MACRF** operand must be coded in the DCB macro instruction.

NTM=<u>absexp</u> (maximum value is 99)

The **NTM** operand specifies the number of tracks to be created in a cylinder index before a higher-level index is created. If the cylinder index exceeds this number, a master index is created by the system; if a master index exceeds this number, the next level of master index is created. The system creates as many as three levels of master indexes. The **NTM** operand is ignored unless the master index option (**OPTCD**=M) is selected.

Source: When an ISAM data set is being created, the **NTM** operand can be supplied in the DCB macro instruction, in the DCB subparameter of a DD statement, or by the problem program before completion of the data control block exit routine. When an ISAM data set is being processed, master index information is supplied to the data control block from the data set label, and the **NTM** operand must be omitted.

OPTCD=[I][L][M][R][U][W][Y]

The **OPTCD** operand specifies the optional services performed by the system when an ISAM data set is being created or updated. The following describes the characters that can be specified (these characters can be specified in any order, and no commas are allowed between characters):

I

specifies that the system uses the independent overflow areas to contain overflow records. Note that it is only the use of the allocated independent overflow area that is optional. Under certain conditions, the system designates an overflow area that was not allocated for independent overflow by the problem program. See "Allocating Space for an Indexed Sequential Data Set" in _Data Administration Guide_.

L

specifies that the data set will contain records flagged for deletion. A record is flagged for deletion by placing a hexadecimal value of 'FF' in the first data byte. Records flagged for deletion remain in the data set until the space is required for another record to be added to the track and are ignored during sequential retrieval of the ISAM data set (QISAM, scan mode). This option cannot be specified for blocked fixed-length records if the relative key position is 0 (**RKP=0**), or it cannot be specified for variable-length records if the relative key position is 4 (**RKP=4**).

When an ISAM data set is being processed with BISAM, a record with a duplicate key can be added to the data set (WRITE KN macro instruction), only when **OPTCD=L** has been specified and the original record (the one whose key is being duplicated) has been flagged for deletion.

M

specifies that the system create and maintain a master index(es) according to the number of tracks specified in the **NTM** operand.

R

specifies that the system place reorganization statistics in the DCBRORG1, DCBRORG2, and DCBRORG3 fields of the data control block. The problem program can analyze these statistics to determine when to reorganize the data set. If the **OPTCD** operand is omitted, the reorganization statistics are automatically provided. However, if the **OPTCD** operand is supplied, **OPTCD=R** must be specified to obtain the reorganization statistics.

U

specifies that the system is to accumulate track index entries in storage and write them as a group for each track of the track index. **OPTCD=U** can be specified only for fixed-length records. The entries are written in fixed-length unblocked format.

W

specifies that the system is to perform a validity check on each record written.

Y

specifies that the system is to use the cylinder overflow area(s) to contain overflow records. If **OPTCD=Y** is specified, the **CYLOFL** operand specifies the number of tracks to be used for the cylinder overflow area. The reserved cylinder overflow area is not used unless **OPTCD=Y** is specified.

Source: When an ISAM data set is created, the **OPTCD** operand can be supplied in the DCB macro instruction, in the DCB subparameter of a DD statement, or by the problem program before an OPEN macro instruction is issued to open the data set. However, all optional services must be requested from the same source. When an existing ISAM data set is processed, the optional service information is supplied to the data control block from the data set label, and the **OPTCD** operand should be omitted.

RECFM={V[B]|F[B]}

The **RECFM** operand specifies the format and characteristics of the records in the data set. If the **RECFM** operand is omitted, variable-length records (unblocked) are assumed. The following describes the characters that can be specified:

B

specifies that the data set contains blocked records.

F

specifies that the data set contains fixed-length records.

V

specifies that the data set contains variable-length records.

Source: When an ISAM data set is created, the **RECFM** operand can be supplied in the DCB macro instruction, in the DCB subparameter of a DD statement, or by the problem program before an OPEN macro instruction is issued to open the data set. When an existing ISAM data set is processed, the record format information is supplied by the data set label, and the RECFM operand should be omitted.

If the record format information is supplied in the DD statement or the DCB, it must agree with the information in the data set label.

RKP=absexp

The **RKP** operand specifies the relative position of the first byte of the key within each logical record. For example, if RKP=9 is specified, the key starts in the 10th byte of the record. The delete option (**OPTCD=L**) should not be specified if the relative key position is the first byte of a blocked fixed-length record or the fifth byte of a variable-length record. If the **RKP** operand is omitted, **RKP=0** is assumed.

If unblocked fixed-length records with **RKP=0** are used, the key is not written as a part of the data record, and the delete option can be specified. If blocked fixed-length records are used, the key is written as part of each data record; either **RKP** must be greater than zero or the delete option must not be used.

If variable-length records (blocked or unblocked) are used, and if the delete option is not specified, **RKP** must be 4 or greater; if the delete option is specified, **RKP** must be specified as 5 or greater. The 4 additional bytes allow for the block descriptor word in variable-length records.

Source: When an ISAM data set is created, the **RKP** operand can be supplied in the DCB macro instruction, in the DCB subparameter of a DD statement, or by the problem program before completion of the data control block exit routine. When an existing ISAM data set is processed, the **RKP** information is supplied by the data set label and the **RKP** operand should be omitted.

SYNAD=relexp

The **SYNAD** operand specifies the address of the error analysis routine given control when an uncorrectable input/output error occurs. The contents of the registers when the error analysis routine is given control are described in Appendix A, "Status Information Following an Input/Output Operation" on page 192.

The error analysis routine must not use the save area pointed to by register 13, because this area is used by the system. The system does not restore registers when it regains control from the error analysis routine. The error analysis routine can issue a RETURN macro instruction that uses the address in register 14 to return control to the system. When control is returned in this manner, the system returns control to the problem program and proceeds as though no error had been encountered; if the error analysis routine continues processing, the results may be unpredictable.

For additional information on error analysis routine processing for indexed sequential data sets, see Data Administration Guide.

Source: The **SYNAD** operand can be supplied in the DCB macro instruction or by the problem program. The problem program can also change the error analysis routine address at any time.

DCB—CONSTRUCT A DATA CONTROL BLOCK (QSAM)

The data control block for the queued sequential access method (QSAM) is constructed during assembly of the problem program. The **DSORG** and **MACRF** operands must be coded in the DCB macro instruction, but the other DCB operands can be supplied to the data control block from other sources. Each DCB operand description contains a heading, "Source." The information under this heading describes the sources from which the operand can be supplied.

For information on additional operands for the DCB macro for the IBM 3890 Document Processor, see _IBM 3890 Document Processor Machine and Programming Description_.

The DCB macro for QSAM is written:

```
[symbol]   DCB        [BFALN={F|D}]
                      [,BFTEK={S|A}]
                      [,BLKSIZE=absexp]
                      [,BUFCB=relexp]
                      [,BUFL=absexp]
                      [,BUFNO=absexp]
                      [,BUFOFF={absexp|L}]
                      [,DDNAME=symbol]¹
                      [,DEVD={{DA}
                             {TA
                              [,DEN={1|2|3|4}]
                              [,TRTCH={C|E|ET|T}]}
                             {PR
                              [,PRTSP={0|1|2|3}]}
                             {PC
                              [,MODE=[C|E][R]]
                              [,STACK={1|2}]
                              [,FUNC={I|P|PW[XT]|R|RP[D]|
                                     RW[T]|RWP[XT][D]|W[T]}]}
                             {RD
                              [,MODE=[C|E][O|R]]
                              [,STACK={1|2}]
                              [,FUNC={I|P|PW[XT]|R|RP[D]|
                                     RW[T]|RWP[XT][D]|W[T]}]}}]
                      ,DSORG={PS|PSU}
                      [,EODAD=relexp]
                      [,EROPT={ACC|SKP|ABE}]
                      [,EXLST=relexp]
                      [,LRECL={absexp|X|0K|nnnnnK}]
                      ,MACRF={{(G{M|L|T|D}[C])}
                             {(P{M|L|T|D}[C])}
                             {(G{M|L|T|D}[C],P{M|L|T|D}[C])}}
                      [,OPTCD={{B}
                              {T}
                              {U[C]}
                              {C[T][B][U]}
                              {H[Z][B]}
                              {J[C][U]}
                              {W[C][T][B][U]}
                              {Z[C][T][B][U]}
                              {Q[C][B][T]}
                              {Z}}]
                      [,RECFM={{U[T][A|M]}
                              {V[B[S][T]|S[T]|T][A|M]}
                              {D[B[S]|[S][A]]}
                              {F[B|S|T|BS|BT][A|M]}}]
                      [,SYNAD=relexp]
```

¹ This parameter must be supplied before an OPEN macro is issued for this DCB; it cannot be supplied in the open exit routine.

The following describes the operands that can be specified in the DCB macro instruction for a QSAM data set:

BFALN={F|D}
The **BFALN** operand specifies the boundary alignment of each buffer in the buffer pool when the buffer pool is constructed automatically or by a GETPOOL macro instruction. If the **BFALN** operand is omitted, the system provides doubleword alignment for each buffer.

If the data set being created or processed contains ISCII/ASCII tape records with a block prefix, the block prefix is entered at the beginning of the buffer, and data alignment depends on the length of the block prefix. For a description of how to specify the block prefix length, refer to the description of the **BUFOFF** operand.

The following describes the characters that can be specified:

F

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

D

specifies that each buffer is on a doubleword boundary.

If the BUILD macro instruction is used to construct the buffer pool, the problem program must control buffer alignment.

Source: The **BFALN** operand can be supplied in the DCB macro instruction, in the DCB subparameter of a DD statement, or by the problem program before completion of the data control block exit routine. If both the **BFALN** and **BFTEK** operands are specified, they must be supplied from the same source.

BFTEK={S|A}

The **BFTEK** operand specifies the buffering technique that is used when the QSAM data set is created or processed. If the **BFTEK** operand is omitted, simple buffering is assumed. The following describes the characters that can be specified:

S

specifies that simple buffering is used.

A

specifies that a logical record interface is used for variable-length spanned records. When **BFTEK=A** is specified, the open routine acquires a record area equal to the length specified in the **LRECL** field plus 32 additional bytes for control information. **LRECL=0** is invalid. The **LRECL** provided at open should be the maximum length in bytes. The open routine uses this value to acquire the record area. When a logical record interface is requested, the system uses the simple buffering technique.

BFTEK=A is invalid with MOVE mode.

To use the simple technique efficiently, the user should be familiar with the three transmittal modes for QSAM and the buffering techniques described in Data Administration Guide.

Source: The **BFTEK** operand can be supplied in the DCB macro instruction, in the DCB subparameter of a DD statement, or by the problem program before completion of the data control block exit routine. If both the **BFTEK** and **BFALN** operands are specified, they must be supplied from the same source.

BLKSIZE=absexp(maximum value is 32760 for IBM standard labels)

The **BLKSIZE** operand specifies the length, in bytes, of a data block for fixed-length records, or it specifies the maximum length, in bytes, of a data block for variable-length or undefined-length records.

The actual value that can be specified in the **BLKSIZE** operand depends on the device type and record format being used. Device capacities are shown in Appendix C, "Device Capacities" on page 194. (For additional information about device capacity, refer to Data Administration Guide.)

For direct access devices when track overflow is used or variable-length spanned records are being processed, the **BLKSIZE** operand can be up to the maximum value. For other record formats used with direct access devices, the value specified in the **BLKSIZE** operand cannot exceed the capacity of a single track.

Because QSAM provides a logical record interface, the device capacities shown in Appendix C, "Device Capacities" on page 194, also apply to a maximum-length logical record. One exception to the device capacity for a logical record is the size of variable-length spanned records. Their length can exceed the value specified in the **BLKSIZE** operand (see the description of the **LRECL** operand).

If fixed-length records are used, the value specified in the **BLKSIZE** operand must be an integral multiple of the value specified in the **LRECL** operand. If the records are unblocked fixed-length records, the value specified in the **BLKSIZE** operand must equal the value specified in the **LRECL** operand if the **LRECL** operand is specified.

If variable-length records are used, the value specified in the **BLKSIZE** operand must include the data length (up to 32756 bytes) plus 4 bytes required for the block descriptor word (BDW). For format-D variable-length records, the minimum **BLKSIZE** is 18 bytes. The maximum is 2048 bytes. For more information about the **BLKSIZE** restrictions, see Data Administration Guide.

If ISCII/ASCII tape records with a block prefix are processed, the value specified in the **BLKSIZE** operand must also include the length of the block prefix. If an ISCII/ASCII format DB or DBS tape data set is opened for output using QSAM with the system acquiring the buffers and BUFOFF=0 specified, the value specified in the **BLKSIZE** operand must be increased by 4 to allow for a 4 byte QSAM internal processing area. If BUFL is specified, the **BUFL** operand value must be increased by 4, instead of the **BLKSIZE** operand value.

If variable-length spanned records are used, the value specified in the **BLKSIZE** operand can be the best one for the device being used or the processing being done. When unit record devices (card or printer) are used, the system assumes records are unblocked; the value specified for the **BLKSIZE** operand is equivalent to one print line or one card. A logical record that spans several blocks is written one segment at a time.

If undefined-length records are used, the problem program can insert the actual record length into the **DCBLRECL** field. See the description of the **LRECL** operand.

Source: The **BLKSIZE** operand can be supplied in the DCB macro instruction, in the DCB subparameter of a DD statement, by the problem program before completion of the data control block exit routine, or by the data set label of an existing data set.

Note: The maximum block size for Version 3 ISO/ANSI/FIPS tapes (ISO 1001-1979 or ANSI X3.27 1978) is 2048 bytes. An attempt to exceed 2048 bytes for a Version 3 tape results in a label validation installation exit being taken.

BUFCB=relexp

The **BUFCB** operand specifies the address of the buffer pool control block constructed by a BUILD or BUILDRCD macro instruction.

If the buffer pool is constructed automatically or by a GETPOOL macro instruction, the system places the address of the buffer pool control block into the data control block, and the **BUFCB** operand should be omitted.

Source: The **BUFCB** operand can be supplied in the DCB macro instruction or by the problem program before completion of the data control block exit routine.

BUFL=absexp (maximum value is 32760)

The **BUFL** operand specifies the length, in bytes, of each buffer in the buffer pool when the buffer pool is acquired automatically. If the **BUFL** operand is omitted, the system acquires buffers with a length equal to the value specified in the **BLKSIZE** operand; if the problem program requires larger buffers, the **BUFL** operand is required. If the data set is for card image mode, the **BUFL** operand is specified as 160 bytes. The description of the **DEVD** operand contains a description of card image mode.

If the data set contains ISCII/ASCII tape records with a block prefix, the value specified in the **BUFL** operand must also include the length of the block prefix. If an ISCII/ASCII format DB or DBS tape data set is opened for output using QSAM and BUFOFF=0 is specified, then the **BUFL** operand value, if specified, must be increased by 4 to allow for a 4-byte QSAM internal processing area.

If the buffer pool is constructed by a BUILD, BUILDRCD, or GETPOOL macro instruction, the **BUFL** operand is not required.

Source: The **BUFL** operand can be supplied in the DCB macro instruction, in the DCB subparameter of a DD statement, or by the problem program before completion of the data control block exit routine.

BUFNO=absexp (maximum value is 255)

The **BUFNO** operand specifies the number of buffers in the buffer pool constructed by a BUILD or BUILDRCD macro instruction, or it specifies the number of buffers to be acquired automatically. If chained scheduling is specified, the value of **BUFNO** determines the maximum number of channel program segments that can be chained and must be specified as more than one. If the **BUFNO** operand is omitted and the buffers are acquired automatically, the system acquires three buffers if the device is a 2540 device or five buffers for any other device type.

If the buffer pool is constructed by a GETPOOL macro instruction, the **BUFNO** operand is not required.

Source: The **BUFNO** operand can be supplied in the DCB macro instruction, in the DCB subparameter of a DD statement, or by the problem program before completion of the data control block exit routine.

BUFOFF={absexp**|L}**

The **BUFOFF** operand specifies the length, in bytes, of the block prefix used with ISCII/ASCII tape data sets. When QSAM is used to read ISCII/ASCII tape records, only the data portion (or its address) is passed to the problem program; the block prefix is not available to the problem program. Block prefixes (except **BUFOFF=L**) cannot be included in QSAM output records. The following can be specified in the **BUFOFF** operand:

absexp

specifies the length, in bytes, of the block prefix. This value can be from 0 to 99 for an input data set. The value must be 0 for writing an output data set with fixed-length or undefined-length records.

L

specifies that the block prefix is 4 bytes long and contains the block length. **BUFOFF=L** is used when format-D records (ISCII/ASCII) are processed. QSAM uses the 4 bytes as a block-descriptor word (BDW). For further information on this operand, see "Variable-Length Records—Format D" in Data Administration Guide.

Source: The **BUFOFF** operand can be supplied in the DCB macro instruction, in the DCB subparameter of a DD statement, or by the problem program before an OPEN macro instruction is issued to open the data set. BUFOFF=absexp can also be supplied by the second system label of an existing data set; BUFOFF=L cannot be supplied by the label of an existing data set.

DDNAME=symbol
The **DDNAME** operand specifies the name used to identify the job control language data definition (DD) statement that defines the data set being created or processed.

Source: The **DDNAME** operand can be supplied in the DCB macro instruction or by the problem program before an OPEN macro instruction is issued to open the data set.

DEVD={DA|TA|PR|PC|RD}[,options]
The **DEVD** operand specifies the device type where the data set can or does reside. The device types above are shown with the optional operand(s) that can be coded when a particular device is used. The devices are listed in order of device independence. For example, if **DEVD=DA** is coded in a DCB macro instruction (or the **DEVD** operand is omitted, which causes a default to **DA**), the data control block constructed during assembly could later be used for any of the other devices, but, if **DEVD=RD** is coded, the data control block can be used only with a card reader or card reader punch. Unless you are certain that device interchangeability is not required, you should either code **DEVD=DA** or omit the operand and allow it to default to **DA**.

If system input is directed to an intermediate storage device, the **DEVD** operand is omitted, and the job control language for the problem program must designate the system input to be used. Similarly, if system output is directed to an intermediate storage device, the **DEVD** operand is omitted, and the job control language for the problem program must designate the system output to be used. If DEVD=PR, PC, or RD is coded, the DCB macro should not be coded within the first 16 bytes of addressability for the control section.

The **DEVD** operand is discussed below according to individual device type:

DEVD=DA
specifies that the data control block can be used for a direct access device (or any of the other device types described following **DA**).

DEVD=TA
[,DEN={1|2|3|4}]
[,TRTCH={C|E|ET|T}]
specifies that the data control block can be used for a magnetic tape data set (or any of the other device types described following **TA**). If **TA** is coded, the following optional operands can be coded:

DEN={1|2|3|4}
The **DEN** operand specifies the recording density in the number of bits-per-inch per track as shown in the following:

Recording Density

DEN	7-Track	9-Track	18-Track
1	556	N/A	N/A
2	800	800 (NRZI)[1]	N/A
3	N/A	1600 (PE)[2]	N/A
4	N/A	6250 (GCR)[3]	N/A

[1] NRZI is for nonreturn-to-zero inverted mode.

[2] PE is for phase encoded mode.

[3] GCR is for group coded recording mode.

TRTCH={C|E|ET|T}
The **TRTCH** operand specifies the recording technique for 7-track tape. One of the above character combinations can be coded. If the **TRTCH** operand is omitted, odd parity with no translation or conversion is assumed. The following describes the characters that can be specified:

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

E
specifies even parity with no translation or conversion.

ET
specifies even parity with BCDIC to EBCDIC translation required, but no data-conversion feature.

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

DEVD=PR
[,PRTSP={0|<u>1</u>|2|3}]
Specifies that the data control block is used for an on-line printer (or any of the other device types following PR). If **PR** is coded, the following optional operand can be coded:

PRTSP={0|1|2|3}
The **PRTSP** operand specifies the line spacing on the printer. This operand is not valid if the **RECFM** operand specifies either machine (**RECFM**=M), ANSI (**RECFM**=A), or ISO control characters. If the **PRTSP** operand is not specified from any source, one is assumed. The following describes the characters that can be specified:

0
specifies that spacing is suppressed (no space).

<u>1</u>
specifies single spacing.

2
specifies double spacing (one blank line between printed lines).

3
specifies triple spacing (two blank lines between printed lines).

Note: MODE and FUNC subparameters cannot be used with this specification.

DEVD=PC
[,MODE=[C|<u>E</u>][R]]
[,STACK={<u>1</u>|2}]
[,FUNC={I|P|PW[XT]|R|RP[D]|RW[T]|RWP[XT][D]|W[T]}]
Specifies that the data control block is used for a card punch (or any of the other device types following

PC). If **PC** is coded, the following optional operands can be specified:

MODE=[C|E][R]]
The **MODE** operand specifies the mode of operation for the card punch. If the **MODE** operand is omitted, **E** is assumed. The following describes the characters that can be specified:

C
specifies that the cards are to be punched in card image mode. In card image mode, the 12 rows in each card column are punched from two consecutive bytes of virtual storage. Rows 12 through 3 are punched from the 6 low-order bits of one byte, and rows 4 through 9 are punched from the 6 low-order bits of the following byte.

E
specifies that cards are to be punched in EBCDIC code.

STACK={1|2}
The **STACK** operand specifies the stacker bin where the card is placed after punching is completed. If this operand is omitted, stacker number 1 is used. The following describes the characters that can be specified:

1
specifies stacker number 1.

2
specifies stacker number 2.

FUNC={I|P|PW[XT]|R|RP[D]|RW[T]|RWP[XT][D]|W[T]}
The **FUNC** operand defines the type of 3525 card punch data sets that is to be used. If the **FUNC** operand is omitted from all sources, a data set opened for input defaults to read only, and a data set opened for output defaults to punch only. The following describes the characters that can be specified in the **FUNC** operand:

D
specifies that the data protection option is to be used. The data protection option prevents punching information into card columns that already contain data. When the data protection option is used, an 80-byte data protection image (DPI) must have been previously stored in SYS1.IMAGELIB. Data protection applies only to the output punch portion of a read and punch or read, punch, and print operation.

I
specifies that the data in the data set is to be punched into cards and printed on the cards; the first 64 characters are printed on line 1 of the card and the remaining 16 characters are printed on line 3.

P
specifies that the data set is for punching cards. See the description of the character X for associated punch and print data sets.

R
specifies that the data set is for reading cards.

T

specifies that the two-line option is used. The two-line print option allows two lines of data to be printed on the card (lines 1 and 3). If T is not specified, the multiline print option is used; this allows printing on all 25 possible print lines. In either case, the data printed may be the same as the data punched in the card, or it may be entirely different data.

W

specifies that the data set is for printing. See the description of the character X for associated punch and print data sets.

X

specifies that an associated data set is opened for output for both punching and printing. Coding the character X is used to distinguish the 3525 printer output data set from the 3525 punch output data set.

Note: If data protection is specified, the data protection image (DPI) must be specified in the FCB subparameter of the DD statement for the data set.

```
DEVD=RD
  [,MODE=[C|E][O|R]]
  [,STACK={1|2}]
  [,FUNC={I|P|PW[XT]|R|RP[D]|RW[T]|RWP[XT][D]|W[T]}]
```

RD

specifies that the data control block is used with a card reader or card read punch. If RD is specified, the data control block cannot be used with any other device type. When RD is coded, the following optional operands can be specified:

MODE=[C|E][O|R]

The MODE operand specifies the mode of operation for the card reader. The following describes the characters that can be specified:

C

specifies that the cards to be read are in card image mode. In card image mode, the 12 rows of each card column are read into two consecutive bytes of virtual storage. Rows 12 through 3 are read into the 6 low-order bits of one byte, and rows 4 through 9 are read into the 6 low-order bits of the following byte.

E

specifies that the cards to be read contain data in EBCDIC code.

O

specifies that the program runs in optical mark read mode (3505 card reader).

R

specifies that the program runs in read-column-eliminate mode (3505 card reader and 3525 card punch, read feature).

Note: If the **MODE** operand for a 3505 or 3525 is specified in the DCB subparameter of a DD statement, either **C** or **E** must be specified if **R** or **O** is specified.

STACK={1|2}

The **STACK** operand specifies the stacker bin into which the card is placed after reading is completed. If this operand is omitted, stacker number 1 is used. The following describes the characters that can be specified:

1
: specifies stacker number 1.

2
: specifies stacker number 2.

FUNC={I|P|PW[XT]|R|RP[D]|RW[T]|RWP[XT][D]|W[T]}

The **FUNC** operand defines the type of 3525 card punch data sets that are used. If the **FUNC** operand is omitted from all sources, a data set opened for input defaults to read only, and a data set opened for output defaults to punch only. The following describes the characters that can be specified in the **FUNC** operand:

D
: specifies that the data protection option is to be used. The data protection option prevents punching information into card columns that already contain data. When the data protection option is used, an 80-byte data protection image (DPI) must have been previously stored in SYS1.IMAGELIB. Data protection applies only to the output punch portion of a read and punch or read, punch, and print operation.

I
: specifies that the data in the data set is to be punched into cards and printed on the cards; the first 64 characters are printed on line 1 of the card and the remaining 16 characters are printed on line 3.

P
: specifies that the data set is for punching cards. See the description of the character X for associated punch and print data sets.

R
: specifies that the data set is for reading cards.

T
: specifies that the two-line option is used. The two-line print option allows two lines of data to be printed on the card (lines 1 and 3). If **T** is not specified, the multiline print option is used; this allows printing on all 25 possible print lines. In either case, the data printed may be the same as the data punched in the card, or it may be entirely different data.

W
: specifies that the data set is for printing. See the description of the character X for associated punch and print data sets.

X
: specifies that an associated data set is opened for output for both punching and printing. Coding the character X is used to distinguish the 3525 printer output data set from the 3525 punch output data set.

Note: If data protection is specified, the data protection image (DPI) must be specified in the FCB subparameter of the DD statement for the data set.

Source: The **DEVD** operand can be supplied only in the DCB macro instruction. However, the optional operands can be supplied in the DCB macro instruction, the DCB subparameter of a DD statement, or by the problem program before completion of the data control block exit routine.

DSORG={PS|PSU}
The **DSORG** operand specifies the organization of the data set and indicates if the data set contains any location-dependent information that would make it unmovable. The following can be specified in the DSORG operand:

PS
Specifies a physical sequential data set.

PSU
Specifies a physical sequential data set that contains location-dependent information.

Source: The **DSORG** operand must be coded in the DCB macro instruction.

EODAD=<u>relexp</u>
The **EODAD** operand specifies the address of the routine given control when the end of an input data set is reached. Control is given to this routine when a GET macro instruction is issued and there are no additional records to be retrieved. If the record format is **RECFM=FS** or **FBS** the end-of-data condition is sensed when file mark is read or if more data is requested after reading a truncated block. If the end of the data set has been reached but no EODAD address has been supplied to the data control block, or if a GET macro instruction is issued after an end-of-data exit is taken, the task is abnormally terminated. For additional information on the EODAD routine, see <u>Data Administration Guide</u>.

Source: The **EODAD** operand can be supplied in the DCB macro instruction or by the problem program before the end of the data set has been reached.

EROPT={ACC|SKP|<u>ABE</u>}
The **EROPT** operand specifies the action taken by the system if an uncorrectable input/output data validity error occurs and no error analysis (SYNAD) routine address has been provided, or it specifies the action taken by the system after the error analysis routine has returned control to the system with a RETURN macro instruction. The specified action is taken for input operations for all devices or for output operations to a printer.

Uncorrectable input/output errors resulting from channel operations or direct access operations that make the next record inaccessible cause the task to be abnormally terminated regardless of the action specified in the **EROPT** operand.

ACC
specifies that the problem program accepts the block causing the error. This action can be specified when a data set is opened for INPUT, RDBACK, UPDAT, or OUTPUT (OUTPUT applies to printer data sets only).

SKP
specifies that the block that caused the error is to be skipped. Specifying **SKP** also causes the buffer

associated with the data block to be released. This action can be specified when a data set is opened for INPUT, RDBACK, or UPDAT.

ABE
: specifies that the error is to result in the abnormal termination of the task. This action can be specified when the data set is opened for INPUT, OUTPUT, RDBACK, or UPDAT.

If the **EROPT** operand is omitted, the **ABE** action is assumed.

Note: If the EROPT operand is ACC or SKIP, accept or skip processing is done after returning from the error analysis (SYNAD) routine. For this reason, FEOV should not be issued from within the error analysis routine.

Source: The **EROPT** operand can be specified in the DCB macro instruction, in the DCB subparameter of a DD statement, or by the problem program at any time. The problem program can also change the action specified at any time.

EXLST=relexp
The **EXLST** operand specifies the address of the problem program exit list. The **EXLST** operand is required if the problem program requires additional processing for the following: user labels, user totaling, data control block exit routines, end-of-volume, block count exits, defining a forms control buffer (FCB) image, using the JFCBE exit (for the 3800 printer), or using the DCB abend exit for abend condition analysis.

For the format and requirements of exit list processing, see Appendix D, "DCB Exit List Format and Contents" on page 197. For additional information about exit routine processing, see Data Administration Guide.

Source: The **EXLST** operand can be supplied in the DCB macro instruction or by the problem program any time before the exit is required by the problem program.

LRECL={absexp**|X|0K|**nnnnn**K}**
The **LRECL** operand specifies the length, in bytes, for fixed-length logical records, or it specifies the maximum length, in bytes, for variable-length or undefined-length (output only) logical records. The value specified in the **LRECL** operand cannot exceed the value specified in the **BLKSIZE** operand except when variable-length spanned records are used.

For fixed-length records that are unblocked, the value specified in the **LRECL** operand must be equal to the value specified in the **BLKSIZE** operand. For blocked fixed-length records, the value specified in the **LRECL** operand must be evenly divisible into the value specified in the **BLKSIZE** operand.

For variable-length logical records, the value specified in the **LRECL** operand must include the maximum data length (up to 32752) plus 4 bytes for the record-descriptor word (RDW).

For undefined-length records, the problem program must insert the actual logical record length into the DCBLRECL field before writing the record, or the maximum-length record will be written.

For variable-length spanned records, the logical record length (**LRECL**) can exceed the value specified in the **BLKSIZE** operand, and a variable-length spanned record can exceed the maximum block size (32760 bytes). When the logical record length exceeds the maximum block size (for non-XLRI processing), **LRECL=X** must be specified and GET or PUT locate mode must be used.

For ISO/ANSI/FIPS variable-length spanned records (RECFM=DS or RECFM=DBS), the extended logical record interface (XLRI) may be used when the maximum logical record length exceeds 32760 bytes. XLRI must be invoked by specifying LRECL=0K or LRECL=nnnnnK. The value nnnnnK may range from 1K to 16383K. The value determines the size of the record area (in 1024-byte units) required to contain the longest logical record of the data set. When LRECL=0K is specified, the length of the longest logical record must come from the DD statement or the data set label. XLRI processing is only valid in QSAM locate mode. You must not specify LRECL=X for RECFM=DS or DBS.

Source: The **LRECL** operand can be supplied in the DCB macro instruction, in the DCB subparameter of a DD statement, by the problem program before completion of the data control block exit routine, or by the data set label of an existing data set. The label will indicate a logical record length of '99999' when an IBM standard label tape contains a logical record equal to or greater than 100K bytes. The label will indicate '00000' if the same maximum is reached for an ISO/ANSI/FIPS label tape.

Note: When LRECL=0K is used in the DCB, the LRECL data must come from JCL, the file label (for an input data set), or from the DCB exit during open merge.

Although LRECL=0K is only valid when RECFM=DS or DBS, you can specify the 0K option on the DCB macro even though the RECFM is not determined until the DCB is opened. (The RECFM is obtained from the data set label or the DD statement.) If you specify neither the DS nor the DBS option, the system turns the 0K indicator off, and restores it when the DCB is closed.

MACRF={{(G{M|L|D}[C])}
{(P{M|L|D}[C])}}
{(G{M|L|D}[C],P{M|L|T|D}[C])}}

The **MACRF** operand specifies the type of macro instructions (GET, PUT or PUTX, CNTRL, RELSE, and TRUNC) and the transmittal modes (move, locate, and data) that are used with the data set being created or processed. The operand can be coded in any of the combinations shown above; the following describes the characters that can be coded:

C

specifies that the CNTRL macro instruction is used with the data set. If the CNTRL macro instruction is specified, the data set should be for a card reader (stacker selection) or printer (carriage and spacing control). The CNTRL option can be specified with GET in the move mode only. Use of the CNTRL macro is invalid for 3525 input data sets.

D

specifies that the data transmittal mode is used (only the data portion of a record is moved to or from the work area). Data mode is used only with variable-length spanned records.

G

specifies that GET macro instructions are used. Specifying **G** also provides the routines that allow the problem program to issue RELSE macro instructions.

L

specifies that the locate transmittal mode is used; the system provides the address of the buffer containing the data.

M

specifies that the move transmittal mode is used; the system moves the data from the buffer to the work area in the problem program.

P

specifies that PUT or PUTX macro instructions are used. Specifying **P** also provides the routines that allow the problem program to issue TRUNC macro instructions.

Note: For data sets processed by QSAM using **MACRF=(GM)** or **MACRF=(PM)**, do not code **BFTEK=A**.

Source: The **MACRF** operand can be supplied only in the DCB macro instruction.

```
OPTCD={{B}
       {T}
       {U[C]}
       {C[T][B][U]}
       {H[Z][B]}
       {J[C][U]}
       {W[C][T][B][U]}
       {Z[C][T][B][U]}
       {Q[C][B][T]}
       {Z}}
```

The **OPTCD** operand specifies the optional services used with the QSAM data set. Two of the optional services, **OPTCD=B** and **OPTCD=H**, cannot be specified in the DCB macro instruction. They are requested in the DCB subparameter of a DD statement. Because all optional services codes must be supplied by the same source, the **OPTCD** operand must be omitted from the DCB macro instruction if either of these options is requested in a DD statement. The following describes the characters that can be specified:

C

requests that chained scheduling be used. **OPTCD=C** cannot be specified when either **BFTEK=A** or **BFTEK=R** is specified for the same data control block. Also, chained scheduling cannot be specified for associated data sets or printing on a 3525 and is ignored for direct access devices.

Note: Except where it is not allowed, chained scheduling is used whether requested or not. For conditions under which it is not allowed, see Data Administration Guide.

J

specifies that the first data byte in the output data line will be a 3800 table reference character. This table reference character selects a particular character arrangement table for the printing of the data line and can be used singly or in conjunction with ISO/ANSI/FIPS or machine control characters. This option is valid only for the IBM 3800 Printing Subsystem. For information on the table reference character and character arrangement table, see IBM 3800 Printing Subsystem Programmer's Guide.

Q

requests that ISCII/ASCII tape records in an input data set be converted to EBCDIC code when the input record has been read, or an output record in EBCDIC code be converted to ISCII/ASCII code before the record is written. For further information on this conversion, see "Variable-Length Records—Format D" in Data Administration Guide.

The Q option is unconditionally set by open routines if the data set is for a tape with ISO/ANSI/FIPS labels. For ISCII/ASCII to EBCDIC or EBCDIC to

ISCII/ASCII translations, see _Magnetic Tape Labels and File Structure Administration_.

T

requests the user totaling function. If this function is requested, the EXLST operand should specify the address of an exit list to be used. **T** cannot be specified for a SYSIN or SYSOUT data set.

U

is specified only for a printer with the universal-character-set feature or the IBM 3800 Printing Subsystem. This option unblocks data checks (permits them to be recognized as errors) and allows analysis by the appropriate error analysis routine (SYNAD exit routine). If the **U** option is omitted, data checks are not recognized as errors.

For the IBM Mass Storage System (MSS): U requests window processing to reduce the amount of staging space required to process large sequential data sets on MSS. DSORG must specify physical sequential, allocation must be in cylinders, and type of I/O accessing must be either INPUT only or OUTPUT only.

W

for DASD, specifies that the system is to perform a validity check on each record written on a direct access device. For buffered devices, specifies that device end interrupt is to be given only when a record is physically on the device. By specifying OPTCD=W with buffered devices, you do not benefit from the performance advantage of buffering.

Z

requests, for magnetic tape, input only, the system to shorten its normal error recovery procedure to consider a data check as a permanent I/O error after five unsuccessful attempts to read a record. This option is available only if it is also specified as a SYSGEN option. **OPTCD=Z** is used when a tape is known to contain errors and there is no need to process every record. The error analysis routine (**SYNAD**) should keep a count of permanent errors and terminate processing if the number becomes excessive.

For direct access devices only, the **Z** option is ignored.

Note: The following describes the optional services that can be specified in the DCB subparameter of a DD statement. If either of these options is requested, the complete **OPTCD** operand must be supplied in the DD statement.

B

If **OPTCD=B** is specified in the DCB subparameter of a DD statement, it forces the end-of-volume (EOV) routine to disregard the end-of-file recognition for magnetic tape. When this occurs, the EOV routine uses the number of volume serial numbers to determine end of file. For an input data set on a standard labeled (SL or AL) tape, the EOV routine will treat EOF labels as EOV labels until the volume serial list is exhausted. After all the volumes have been read, control is passed to the user's end-of-data routine. This option allows SL or AL tapes to be read out of volume sequence or to be concatenated to another tape using one DD statement.

H

If **OPTCD=H** is specified in the DCB subparameter of a DD statement, it specifies that the DOS/OS interchange feature is being used with the data set.

```
RECFM={{U[T][A|M]}}
      {V[B[S][T]|S[T]|T][A|M]}
      {D[B[S]|[S][A]]}
      {F[B|S|T|BS|BT][A|M]}}
```

The **RECFM** operand specifies the record format and characteristics of the data set being created or processed. All record formats can be used in QSAM. The following describes the characters that can be specified:

A

specifies that the records in the data set contain ISO/ANSI/FIPS control characters. For a description of control characters, see Appendix E, "Control Characters" on page 199.

B

specifies that the data set contains blocked records.

D

specifies that the data set contains variable-length ISCII/ASCII tape records. See **OPTCD=Q** and the **BUFOFF** operand for a description of how to specify ISCII/ASCII data sets.

F

specifies that the data set contains fixed-length records.

M

specifies that the records in the data set contain machine code control characters. For a description of control characters, see Appendix E, "Control Characters" on page 199. **RECFM=M** cannot be used with ISCII/ASCII data sets.

S

specifies, for fixed-length records, that the records are to be written as standard blocks; the data set does not contain any truncated blocks or unfilled tracks, with the exception of the last block or track in the data set. Do not code S for fixed-length records to retrieve records from a data set that was created using a **RECFM** other than standard.

For variable-length records, S specifies that a record can span more than one block.

T

specifies that track overflow is used with the data set. Track overflow allows a record to be written partially on one track and the remainder of the record on the following track (if required). Chained scheduling (**OPTCD=C**) cannot be requested if track overflow is used.

U

specifies that the data set contains undefined-length records.

Note: Format-U records are not supported for Version 3 ISO/ANSI/FIPS tapes. An attempt to process a format-U record for a Version 3 tape results in a label validation installation exit being taken.

ISO/ANSI Version 1 (ISO 1001-1969 or ANSI X3.27-1969) format-U records can be used for input only. These records are the same as the format-U records described above, except that the control characters must be ISO/ANSI control characters, and block prefixes can be used.

V

specifies that the data set contains variable-length records.

Notes:

- **RECFM=V** cannot be specified for a card reader data set or an ISO/ANSI/FIPS tape data set.
- **RECFM=VS, VBS, DS,** or **DBS** cannot be specified for a SYSIN data set.
- **RECFM=DS** or **RECFM=DBS** provides blocking, unblocking, and segmenting for Version 3 ISO/ANSI/FIPS tape data sets.

Source: The **RECFM** operand can be supplied in the DCB macro instruction, in the DCB subparameter of a DD statement, by the problem program before completion of the data control block exit routine, or by the data set label of an existing data set.

SYNAD=_relexp_

The **SYNAD** operand specifies the address of the error analysis routine given control if an uncorrectable input/output error occurs. The contents of the registers when the error analysis routine is given control are described in Appendix A, "Status Information Following an Input/Output Operation" on page 192.

The error analysis routine must not use the save area pointed to by register 13, because this area is used by the system. The system does not restore registers when it regains control from the error analysis routine. The error analysis routine can issue a RETURN macro instruction that uses the address in register 14 to return control to the system.

If the error condition was the result of a data-validity error, the control program takes the action specified in the **EROPT** operand; otherwise, the task is abnormally terminated. The control program takes these actions when the **SYNAD** operand is omitted or when the error analysis routine returns control.

Source: The **SYNAD** operand can be supplied in the DCB macro instruction or by the problem program. The problem program can also change the error routine address at any time.

DCBD—PROVIDE SYMBOLIC REFERENCE TO DATA CONTROL BLOCKS (BDAM, BISAM, BPAM, BSAM, QISAM, AND QSAM)

The DCBD macro instruction is used to generate a dummy control section that provides symbolic names for the fields in one or more data control blocks. The names and attributes of the fields appear as part of the description of each data control block in Appendix F, "Data Control Block Symbolic Field Names" on page 202. Attributes of the symbolically named fields in the dummy section are the same as the fields in the data control blocks, with the exception of fields containing 3-byte addresses. The symbolically named fields containing 3-byte addresses have length attributes of 4 and are aligned on fullword boundaries.

The labels generated by the DCBD macro should not be defined within a user program. The macro labels are structured as DCB xxxxx, where DCB is the first 3 characters and xxxxx is 1 to 5 alphameric characters.

The name of the dummy control section generated by a DCBD macro instruction is IHADCB. The use of any of the symbolic names provided by the dummy section must be preceded by a USING instruction specifying IHADCB and a dummy section base register (which contains the address of the actual data control block). The DCBD macro instruction can only be issued once within any assembled module; however, the resulting symbolic names can be used for any number of data control blocks by changing the address in the dummy section base register. The DCBD macro instruction can be coded at any point in a control section; if coded at any point other than at the end of a control section, however, the control section must be resumed by coding a CSECT instruction.

The DCBD macro is written:

b	**DCBD**	**[DSORG=({GS| [BS][,DA][,IS][,LR][,PO][,PS][,QS]})] [,DEVD=([DA][,PC][,PR][,RD][,TA] [,MR])]**

DSORG=({GS|[BS][,DA][,IS][,LR][,PO][,PS][,QS]})
The **DSORG** operand specifies the types of data control blocks for which symbolic names are provided. If the **DSORG** operand is omitted, the **DEVD** operand is ignored, and symbolic names are provided only for the 'foundation block' portion that is common to all data control blocks. One or more of the following pairs of characters can be specified (each pair of characters must be separated by a comma):

BS
specifies a data control block for a sequential data set and basic access method.

DA
specifies a data control block for a direct data set.

IS
specifies a data control block for an indexed sequential data set.

LR
specifies a dummy section for the logical record length field (DCBLRECL) only.

PO
specifies a data control block for a partitioned data set.

PS
specifies a data control block for a sequential data set. **PS** includes both **BS** and **QS**.

QS
: specifies a data control block for a sequential data set and queued access method.

GS
: specifies a data control block for graphics; this operand cannot be used in combination with any of the above.

DEVD=[DA][,PC][,PR][,RD][,TA][,MR]
The **DEVD** operand specifies the types of devices on which the data set can reside. If the **DEVD** operand is omitted and a sequential data set is specified in the **DSORG** operand, symbolic names are provided for all the device types listed below. One or more of the following pairs of characters can be specified; each pair of characters must be separated by a comma:

DA
: Direct access device

PC
: On-line punch

PR
: On-line printer

RD
: On-line card reader or read punch feed

TA
: Magnetic tape

MR
: Magnetic character reader

ESETL—END SEQUENTIAL RETRIEVAL (QISAM)

The ESETL macro instruction ends the sequential retrieval of data from an indexed sequential data set and causes the buffers associated with the specified data control block to be released. An ESETL macro instruction must separate SETL macro instructions issued for the same data control block.

The ESETL macro is written:

[symbol]	**ESETL**	dcb address

dcb address—RX-Type Address, (2-12), or (1)
The dcb address operand specifies the address of the data control block opened for the indexed sequential data set being processed.

FEOV—FORCE END OF VOLUME (BSAM AND QSAM)

The FEOV macro instruction causes the system to assume an end-of-volume condition, and causes automatic volume switching. Volume positioning for magnetic tape can be specified by the option operand. If no option is coded, the positioning specified in the OPEN macro instruction is used. Output labels are created as required and new input labels are verified. The standard exit routines are given control as specified in the data control block exit list. For BSAM, all input and output operations must be tested for completion before the FEOV macro instruction is issued. The end-of-data-set (EODAD) routine is given control if an input FEOV macro instruction is issued for the last volume of an input data set. FEOV is ignored if issued for a SYSIN or SYSOUT data set.

The FEOV macro is written:

[symbol]	**FEOV**	dcb address **[,REWIND|,LEAVE]**

dcb address—RX-Type Address, (2-12), or (1)
: The dcb address operand specifies the address of the data control block for an opened sequential data set.

REWIND
: requests that the system position the tape at the load point regardless of the direction of processing.

LEAVE
: requests that the system position the tape at the logical end of the data set on that volume; this option causes the tape to be positioned at a point after the tapemark that follows the trailer labels. Note that multiple tape units must be available to achieve this positioning. If only one tape unit is available, its volume is rewound and unloaded.

Note: If an FEOV macro is issued for a multivolume data set with spanned records that is being read using QSAM, errors may occur when the next GET macro is issued following an FEOV macro if the first segment on the new volume is not the first segment of a record. The errors include duplicate records, program checks in the user program, and invalid input from the variable spanned data set.

The FEOV macro should not be used within the error analysis routine (SYNAD).

FIND—ESTABLISH THE BEGINNING OF A DATA SET MEMBER (BPAM)

The FIND macro instruction causes the system to use the address of the first block of a specified partitioned data set member as the starting point for the next READ macro instruction for the same data set. All previous input and output operations that specified the same data control block must have been tested for completion before the FIND macro instruction is issued.

The FIND macro is written:

[symbol]	**FIND**	dcb address ,{name address,**D**|relative address list,**C**}

dcb address—RX-Type Address, (2-12), or (1)
: The dcb address operand specifies the address of the data control block for the opened partitioned data set being processed.

name address—RX-Type Address, (2-12), or (0)
: The name address operand specifies the address of an 8-byte field that contains the data set member name. The name must start in the first byte and be padded on the right (if necessary) to complete the 8 bytes.

D
: Specifies that only a member name has been supplied, and the access method must search the directory of the data set indicated in the data control block to find the location of the member.

relative address list—RX-Type Address, (2-12), or (0)
: The relative address list operand specifies the address of the area that contains the relative address (TTRK) for the beginning of a data set member. The relative address can be a list entry completed by using a BLDL macro instruction for the data set being processed, or the relative address can be supplied by the problem program.

C
: specifies that a relative address has been supplied, and no directory search is required. The relative address supplied is used directly by the access method for the next input operation.

Note: When using the FIND macro, the DCBRELAD address in the DCB is updated. The FIND macro should not be used after WRITE and STOW processing without first closing the data set and reopening it for INPUT processing.

COMPLETION CODES

For relative address list, **C**, when the system returns control to the problem program, the low-order byte of register 15 contains the following return code; the 3 high-order bytes of register 15 are set to 0.

relative address list, **C**

00 — At all times. If the relative address is in error, execution of the next CHECK macro instruction causes control to be passed to the error analysis (SYNAD) routine.

For name address, **D**, when the system returns control to the problem program, the low-order byte of register 15 contains a return code and the low-order byte of register 0 contains a reason code. The 3 high-order bytes of these two registers are set to 0.

name address, **D**

Return Code (15)	**Reason Code (0)**	**Meaning**
00 (X'00')	00 (X'00')	Successful execution.
04 (X'04')	00 (X'00')	Name not found.
08 (X'08')	00 (X'00')	Permanent I/O error during directory search.
08 (X'08')	04 (X'04')	Insufficient virtual storage available.
08 (X'08')	08 (X'08')	Invalid DEB. (Not in key 0 through 7.)

FREEBUF—RETURN A BUFFER TO A POOL (BDAM, BISAM, BPAM, AND BSAM)

The FREEBUF macro instruction causes the system to return a buffer to the buffer pool assigned to the specified data control block. The buffer must have been acquired using a GETBUF macro instruction.

The FREEBUF macro is written:

[symbol]	**FREEBUF**	dcb address ,register

dcb address—RX-Type Address, (2-12), or (1)
: The dcb address operand specifies the address of the data control block for an opened data set to which the buffer pool has been assigned.

register—(2-12)
: The register operand specifies one of registers 2 through 12 that contains the address of the buffer being returned to the buffer pool.

FREEDBUF—RETURN A DYNAMICALLY OBTAINED BUFFER (BDAM AND BISAM)

The FREEDBUF macro instruction causes the system to return a buffer to the buffer pool assigned to the specified data control block. The buffer must have been acquired through dynamic buffering; that is, by coding **'S'** for the area address operand in the associated READ macro instruction.

Note: A buffer acquired dynamically can also be released by a WRITE macro instruction; see the description of the WRITE macro instruction for BDAM or BISAM.

The FREEDBUF macro is written:

[<u>symbol</u>]	**FREEDBUF**	<u>decb address</u> **,{**K**|D}** ,<u>dcb address</u>

<u>decb address</u>—RX-Type Address, (2-12), or (0)
The decb address operand specifies the address of the data event control block (DECB) used or created by the READ macro instruction that acquired the buffer dynamically.

K
specifies that BISAM is being used.

D
specifies that BDAM is being used.

<u>dcb address</u>—RX-Type Address, (2-12), or (1)
The dcb address operand specifies the address of the data control block for the opened data set being processed.

FREEPOOL—RELEASE A BUFFER POOL (BDAM, BISAM, BPAM, BSAM, QISAM, AND QSAM)

The FREEPOOL macro instruction causes an area of storage, previously acquired for a buffer pool for a specified data control block, to be released. The area must have been acquired either automatically (except when dynamic buffer control is used) or by the execution of a GETPOOL macro instruction. For queued access methods, the FREEPOOL macro instruction must not be issued until after a CLOSE macro instruction has been issued for all the data control blocks using the buffer pool. For basic access methods, the FREEPOOL macro instruction can be issued as soon as the buffers are no longer required. A buffer pool should be released only once, regardless of the number of data control blocks sharing the buffer pool.

The FREEPOOL macro is written:

[symbol]	**FREEPOOL**	dcb address

dcb address—RX-Type Address, (2-12), or (1)
The dcb address operand specifies the address of a data control block to which the buffer pool has been assigned.

GET—OBTAIN NEXT LOGICAL RECORD (QISAM)

The GET macro instruction causes the system to retrieve the next record. Control is not returned to the problem program until the record is available.

The GET macro is written:

[symbol]	GET	dcb address [,area address]

dcb address—RX-Type Address, (2-12), or (1)
The dcb address operand specifies the address of the data control block for the opened input data set being retrieved.

area address—RX-Type Address, (2-12), or (0)
The area address operand specifies the storage address into which the system is to move the record (move mode only). Either the move or locate mode can be used with QISAM, but they must not be mixed within the specified data control block. The following describes operations for move and locate modes:

Locate Mode: If locate mode has been specified in the data control block, the area address operand must be omitted. The system returns the address of the buffer segment containing the record in register 1.

Move Mode: If move mode has been specified in the data control block, the area address operand must specify the address in the problem program into which the system will move the record. If the area address operand is omitted, the system assumes that register 0 contains the area address. When control is returned to the problem program, register 0 contains the area address, and register 1 contains the address of the data control block.

Notes:

1. The end-of-data-set (EODAD) routine is given control if the end of the data set is reached; the data set may be closed if processing is completed, or an ESETL macro must be issued before a SETL macro to continue further input processing.

2. The error analysis (SYNAD) routine is given control if the input operation could not be completed successfully. The contents of the general registers when control is given to the SYNAD exit routine are described in Appendix A, "Status Information Following an Input/Output Operation" on page 192.

3. When the key of an unblocked record is retrieved with the data, the address of the key is returned as follows (see the SETL macro instruction):

   Locate Mode: The address of the key is returned in register 0.

   Move Mode: The key appears in front of the record in your buffer area.

4. If a GET macro instruction is issued for a data set and the previous request issued for the same data set was an OPEN, ESETL, or unsuccessful SETL (no record found), a SETL B (key and data) is invoked automatically, and the first record in the data set is returned.

GET—OBTAIN NEXT LOGICAL RECORD (QSAM)

The GET macro instruction causes the system to retrieve the next record. Various modes are available and are specified in the DCB macro instruction. In the locate mode, the GET macro instruction locates the next sequential record or record segment to be processed. The system returns the address of the record in register 1 and places the length of the record or segment in the logical record length (DCBLRECL) field of the data control block. The DCBLRECL field is not changed when GET is used in XLRI processing. The user can process the record within the input buffer or move the record to a work area.

In move mode, the GET macro instruction moves the next sequential record to the user's work area. This work area must be large enough to contain the largest logical record of the data set and its record-descriptor word (variable-length records). The system returns the address of the work area in register 1. The record length is placed in the DCBLRECL field. Move mode can be used only with simple buffering.

In data mode, which is available only for variable-length spanned records, the GET macro instruction moves only the data portion of the next sequential record to the user's work area. The **TYPE=P** operand cannot be used with data mode.

If the ISCII/ASCII translation routines are included when the operating system is generated, translation can be requested by coding LABEL=(,AL) or (,AUL) in the DD statement, or it can be requested by coding OPTCD=Q in the DCB macro instruction or DCB subparameter of the DD statement. When translation is requested, all QSAM records whose record format (RECFM operand) is F, FB, D, DS, DB, DBS, or U are automatically translated from ISCII/ASCII code to EBCDIC code as soon as the input buffer is full. For translation to occur correctly, all input data must be in ISCII/ASCII code.

The GET macro is written:

[<u>symbol</u>]	**GET**	{<u>dcb address</u>|<u>pdab address</u>} [,<u>area address</u>] **[,TYPE=P]**

<u>dcb address</u>—RX-Type Address, (2-12), or (1)
The dcb address operand specifies the address of the data control block for the opened input data set being retrieved.

<u>pdab address</u>—RX-Type Address, (2-12), or (1)
The pdab address operand specifies the address of the parallel data access block for the opened input data sets from which a record is to be retrieved. When pdab address is used, **TYPE=P** must be coded.

<u>area address</u>—RX-Type Address, (2-12), or (0)
The area address operand specifies the address of an area into which the system is to move the record (move or data mode). The move, locate, or data mode can be used with QSAM, but must not be mixed within the specified data control block. If the area address operand is omitted in the move or data mode, the system assumes that register 0 contains the area address. The following describes the operation of the three modes:

Locate Mode: If locate mode has been specified in the data control block, the area address operand must be omitted. The system returns the address of the beginning buffer segment containing the record in register 1. If the data set is open for RDBACK, register 1 will point to the beginning of the record.

When retrieving variable-length spanned records, and the logical record interface (LRI) or extended logical record interface (XLRI) is not used, the records are obtained one segment at a time. The problem program must retrieve additional segments by issuing subsequent GET macro instructions, except when a logical record interface is requested (by specifying BFTEK=A in the DCB macro instruction or by issuing a BUILDRCD macro instruction, or by specifying DCBLRECL=0K or nnnnnK in the DCB macro). In this case, the control program retrieves all record segments and assembles the segments into a complete logical record. The system returns the address of this record area in register 1.

When the maximum logical record length is greater than 32756 bytes, LRECL=X must be specified in the data control block, and the problem program must assemble the segments into a complete logical record. LRECL=X and/or segment mode processing is invalid for ISO/ANSI/FIPS spanned records, RECFM=DS or RECFM=DBS.

Move Mode: If move mode has been specified in the data control block, the area address operand specifies the beginning address of an area in the problem program into which the system will move the record. If the data set is open for RDBACK, the area address operand specifies the ending address of an area in the problem program.

If move mode has been specified in the data control block, do not code **BFTEK=A**.

For variable-length spanned records, the system constructs the record-descriptor word in the first four bytes of the area and assembles one or more segments into the data portion of the logical record area; the segment descriptor words are removed. When XLRI mode is used, the record descriptor word (RDW) in the record area is a fullword value.

Data Mode: If data mode has been specified in the data control block (data mode can be specified for variable-length spanned records only), the area address operand specifies the address of the area in the problem program into which the system will move the data portion of the logical record; a record-descriptor word is not constructed when data mode is used. The **TYPE=P** operand cannot be used with data mode.

Extended Logical Record Interface (XLRI): When the GET macro is used in XLRI mode, the address returned in register 1 points to a fullword record length value. The three low-order bytes of the fullword indicate the length of the complete logical record plus four bytes for the fullword.

XLRI mode requires a record area to assemble a complete logical record from the segments that are read.

If a record area is not automatically obtained by OPEN processing, you can construct a record by using the BUILDRCD macro before issuing the OPEN. The DCB **LRECL** field indicates the length of the area in K units (1024 bytes) required to contain the longest logical record of the data set.

Note: If spanned records extend across volumes, errors may occur when using the GET macro if a volume that begins with a middle or last record segment is mounted first, or if an FEOV macro is issued followed by a GET macro. QSAM cannot begin reading from the middle of the record. (This applies to move mode, data mode, and locate mode if logical record interface is specified.)

TYPE=P—Coded as shown
The **TYPE=P** and PDAB address operands are used to retrieve a record from a queue of input data sets that have been opened. The open and close routines add and delete DCB addresses in the queue. The DCB from which a record is retrieved can be located from information in the PDAB. For this purpose, the formatting macro, PDABD, should be used.

GET ROUTINE EXITS

The end-of-data-set (EODAD) routine is given control if the end of the data set is reached; the data set must be closed. Issuing a GET macro instruction in the EODAD routine results in abnormal termination of the task.

The error analysis (SYNAD) routine is given control if the input operation could not be completed successfully. The contents of the general registers when control is given to the SYNAD exit routine are described in Appendix A, "Status Information Following an Input/Output Operation" on page 192.

GETBUF—OBTAIN A BUFFER (BDAM, BISAM, BPAM, AND BSAM)

The GETBUF macro instruction causes the control program to obtain a buffer from the buffer pool assigned to the specified data control block and to return the address of the buffer in a designated register. The BUFCB field of the data control block must contain the address of the buffer pool control block when the GETBUF macro instruction is issued. The system returns control to the instruction following the GETBUF macro instruction. The buffer obtained must be returned to the buffer pool using a FREEBUF macro instruction.

The GETBUF macro is written:

[symbol]	**GETBUF**	dcb address ,register

dcb address—RX-Type Address, (2-12), or (1)
: The dcb address operand specifies the address of the data control block that contains the buffer pool control block address.

register—(2-12)
: The register operand specifies one of the registers 2 through 12 in which the system is to place the address of the buffer obtained from the buffer pool. If no buffer is available, the contents of the designated register are set to 0.

GETPOOL—BUILD A BUFFER POOL (BDAM, BISAM, BPAM, BSAM, QISAM, AND QSAM)

The GETPOOL macro instruction causes a buffer pool to be constructed in a storage area acquired by the system. The system places the address of the buffer pool control block in the BUFCB field of the data control block. The GETPOOL macro instruction must be issued either before an OPEN macro instruction is issued or during the data control block exit routine for the specified data control block.

The GETPOOL macro is written:

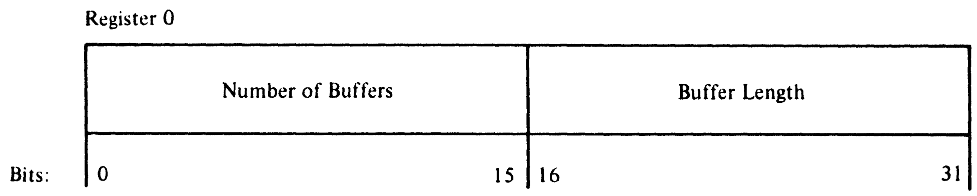
[symbol]	**GETPOOL**	dcb address ,{number of buffers,buffer length|(0)}

dcb address—RX-Type Address, (2-12), or (1)
The dcb address operand specifies the address of the data control block to which the buffer pool is assigned. Only one buffer pool can be assigned to a data control block.

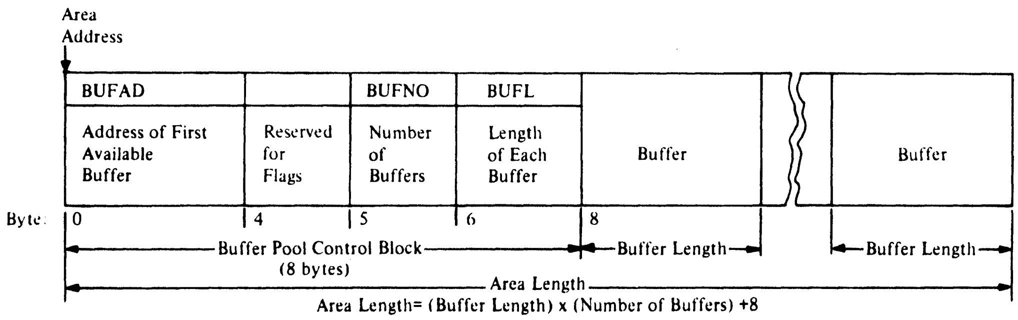
number of buffers—symbol, decimal digit, absexp, or (2-12)
The number-of-buffers operand specifies the number of buffers in the buffer pool to a maximum of 255.

buffer length—symbol, decimal digit, absexp, or (2-12)
The buffer length operand specifies the length, in bytes, or each buffer in the buffer pool. The value specified for the buffer length must be a doubleword multiple; otherwise, the system rounds the value specified to the next higher doubleword multiple. The maximum length that can be specified is 32760 bytes. For QSAM, the buffer length must be at least as large as the value specified in the block size (DCBBLKSI) field in the data control block.

(0)—Coded as shown
The number of buffers and buffer length can be specified in general register 0. If (0) is coded, register 0 must contain the binary values for the number of buffers and buffer length as shown in the following illustration:



The following illustration shows the format of the buffer pool. The buffer pool and the associated storage area are released by issuing a FREEPOOL macro instruction after issuing a CLOSE macro instruction for the data set indicated in the specified data control block.


Area Address
BUFAD
BUFNO
BUFL
Address of First Available Buffer
Reserved for Flags
Number of Buffers
Length of Each Buffer
Buffer
Buffer
Byte: 0
4
5
6
8
Buffer Pool Control Block (8 bytes)
Buffer Length
Buffer Length
Area Length
Area Length= (Buffer Length) x (Number of Buffers) +8

MSGDISP—MESSAGE DISPLAY (DRIVE READY)

The MSGDISP macro is used to load the message display on an IBM 3480 Magnetic Tape Subsystem. Functions for the display include:

- Displaying a ready message
- Mount volume[1]
- Demount volume[1]
- Reset display[1]
- Verify volume[1]
- Generalized display[1]

[1] For an explanation of these MSGDISP macro functions, see System-Data Administration.

MSGDISP—DISPLAYING A READY MESSAGE

The MSGDISP macro is written:

[symbol]	MSGDISP	RDY ,DCB=addr [,TXT={'msgtxt'|addr}]

RDY
specifies that text supplied in the TXT parameter will be displayed in positions 2 through 7 of the display while the data set is open. The display will be steady (not flashing) and will be enclosed in parentheses. The display will also be written to the tape pool console (routing code 3, descriptor code 7).

DCB=addr
specifies the address of a DCB opened to a data set on the mounted volume. If multiple devices are allocated, the message display will be directed to the one containing the volume currently in use.

Note: If multiple devices or multiple volumes are allocated, you may update a message display after an end-of-volume condition by using the EOV exit specified in a DCB exit list. In the case of a concatenated data set with unlike characteristics, the open DCB exit may be used to update the display.

addr—RX-Type address, A-Type address, or (2-12)
specifies an in-storage address of the opened DCB.

TXT={'msgtxt'|addr}
specifies as many as six characters to be displayed in positions 2 through 7. If TXT is not specified, blanks will be displayed.

'msgtxt'
specifies the 1- to 6-character text. The text must be enclosed in apostrophes.

addr—RX-Type address, A-Type address, or (2-12)
specifies an in-storage address of an area containing the text to be displayed.

MSGDISP—LIST FORM

The list form of the MSGDISP macro is written:

[symbol]	MSGDISP	[RDY] [,DCB=addr] ,MF=L [,TXT={'msgtxt'|addr}]

RDY
specifies that text supplied in the TXT parameter will be displayed in positions 2 through 7 while a data set is open. The display will be steady (not flashing) and will be enclosed in parentheses. The display will also be written to the tape pool console (routing code 3, descriptor code 7).

DCB=addr
specifies the address of a DCB opened to a data set on the mounted volume. If multiple devices are allocated, the message display will be directed to the one containing the volume currently in use.

Note: If multiple devices or multiple volumes are allocated, you may update a message display after an end-of-volume condition by using the EOV exit specified in a DCB exit list. In the case of a concatenated data set with unlike characteristics, the open DCB exit may be used to update the display.

addr—A-Type address
specifies an in-storage address of the opened DCB.

MF=L
specifies the list form of MSGDISP. This generates a parameter list that contains no executable instructions. The list can be used as input to and can be modified by the execute form.

TXT={'msgtxt'|addr}
specifies as many as six characters to be displayed in positions 2 through 7. If TXT is not specified, blanks will be displayed.

'msgtxt'
specifies the 1- to 6-character text. The text must be enclosed in apostrophes.

addr—A-Type address
specifies an in-storage address of an area containing the text to be displayed.

MSGDISP—EXECUTE FORM

The execute form of the MSGDISP macro is written:

[symbol]	MSGDISP	RDY [,DCB=addr] ,MF=(E,addr) [,TXT={'msgtxt'|addr}]

RDY
specifies that text supplied in the TXT parameter will be displayed in positions 2 through 7 while a data set is open. The display will be steady (not flashing) and will be enclosed in parentheses. The display will also be written to the tape pool console (routing code 3, descriptor code 7).

DCB=addr
specifies the address of a DCB opened to a data set on the mounted volume. If multiple devices are allocated, the message display will be directed to the one containing the volume currently in use.

Note: If multiple devices or multiple volumes are allocated, you may update a message display after an end-of-volume condition by using the EOV exit specified in a DCB exit list. In the case of a concatenated data set with unlike characteristics, the open DCB exit may be used to update the display.

addr—RX-Type address, A-Type address, or (2-12)
specifies an in-storage address of the opened DCB.

MF=(E,addr)
specifies that the execute form of MSGDISP and an existing parameter list will be used.

addr—RX-Type address, (1), or (2-12)
specifies an in-storage address of the parameter list.

TXT={'msgtxt'|addr}
specifies as many as six characters to be displayed in positions 2 through 7. If TXT is not specified, blanks will be displayed.

'msgtxt'
specifies the 1- to 6-character text. The text must be enclosed in apostrophes.

addr—RX-Type address, A-Type address, or (2-12)
specifies an in-storage address of an area containing the text to be displayed.

COMPLETION CODES

When the system returns control to your problem program, the low-order byte of register 15 contains a return code; the low-order byte of register 0 contains a reason code:

Return Code (15)	Reason Code (0)	Meaning
00 (X'00')		Successful completion
04 (X'04')		Device does not support MSGDISP
08 (X'08')	01 (X'01')	Invalid parameter
08 (X'08')	02 (X'02')	Invalid DCB or DEBCHK error
08 (X'08')	03 (X'03')	Environmental error
08 (X'08')	04 (X'04')	Authorization violation
08 (X'08')	05 (X'05')	Invalid UCB
08 (X'08')	06 (X'06')	Invalid request
08 (X'08')	11 (X'0B')	Unsuccessful ESTAE macro call
08 (X'08')	12 (X'0C')	Unsuccessful GETMAIN request
12 (X'0C')		Input/output error (I/O Supervisor posted the request as an error)

Note: An I/O error occurs for load display if the drive display has a hardware failure.

NOTE—PROVIDE RELATIVE POSITION (BPAM AND BSAM—TAPE AND DIRECT ACCESS ONLY)

The NOTE macro instruction causes the system to return the position of the last block read from or written into a data set. All input and output operations using the same data control block must be tested for completion before the NOTE macro instruction is issued.

The capability of using the NOTE macro instruction is automatically provided when a partitioned data set is used (DSORG=PO or POU), but, when a sequential data set (BSAM) is used, the use of NOTE/POINT macro instructions must be indicated in the MACRF operand of the DCB macro instruction.

The NOTE macro is written:

[symbol]	NOTE	dcb address [,TYPE={ABS|REL}]

dcb address—RX-Type Address, (2-12), or (1)
The dcb address operand specifies the address of the data control block opened for the partitioned or sequential data set being processed.

TYPE={ABS|REL}

ABS

specifies that, after NOTE executes successfully (contents of register 15 is 0), register 0 contains the physical block identifier for the next data block waiting for transfer between main storage and the control unit buffer, and register 1 contains the physical block identifier of the next data block waiting for transfer between the control unit buffer and the tape drive.

If you subtract the low-order 20 bits of register 1 from the low-order 20 bits of register 0, the remainder is the number of data blocks left in the control unit buffer. A negative remainder means the buffer is in read mode, and a positive remainder means the buffer is in either write or read-backward mode. A zero remainder means that no data is buffered.

REL

causes the system to return the relative position of the last block read from or written into a data set. The position, in terms of the current volume, is returned in register 1 as follows:

Magnetic Tape

The block number is in binary, right-adjusted in register 1 with high-order bits set to zero. Do not use a NOTE macro instruction for tapes without standard labels when:

- The data set is opened for RDBACK (specified in the OPEN macro instruction) or
- The DISP parameter of the DD statement for the data set specifies DISP=MOD.

Direct Access Device

TTRz format, where:

TT is a 2-byte relative track number.

R is a 1-byte block (record) number on the track indicated by TT.

z is a byte set to zero.

The NOTE macro instruction cannot be used for SYSOUT data sets.

Note: When a direct access device is being used, the amount of remaining space on the track is returned in register 0 if a NOTE macro instruction follows a WRITE macro instruction; if a NOTE macro instruction follows a READ or POINT macro instruction, the track capacity of the direct access device is returned in register 0.

COMPLETION CODES

When the system returns control to your problem program and you have specified the ABS parameter, the low-order byte of register 15 contains a return code; the low-order byte of register 0 contains a reason code:

Return Code (15)	Reason Code (0)	Meaning
00 (X'00')		Successful completion.
04 (X'04')		Device does not support block identifier.
08 (X'08')	01 (X'01')	Incorrect parameter.
08 (X'08')	02 (X'02')	Incorrect DCB or a DEBCHK error.
08 (X'08')	03 (X'03')	Environmental error.
08 (X'08')	11 (X'0B')	Unsuccessful call to ESTAE macro.
08 (X'08')	12 (X'0C')	Unsuccessful GETMAIN request.
12 (X'0C')		Input/output error.

OPEN—LOGICALLY CONNECT A DATA SET (BDAM, BISAM, BPAM, BSAM, QISAM, AND QSAM)

The OPEN macro instruction causes the specified data control block(s) to be completed and the data set(s) identified in the data control block(s) to be prepared for processing. Input labels are analyzed and output labels are created. Control is given to exit routines as specified in the data control block exit list. The processing method (option 1) is designated to provide correct volume positioning for the data set and define the processing mode (INPUT, OUTPUT, and so forth) for the data set(s). Final volume positioning (when volume switching occurs) can be specified (option 2) to override the positioning implied by the DD statement DISP parameter. Option 2 applies only to volumes in a multivolume data set other than the last volume. Any number of data control block addresses and associated options may be specified in the OPEN macro instruction.

The maximum number of DCBs that can be concurrently open to one unit is 127.

If associated data sets for a 3525 card punch are being opened, all associated data sets must be open before an I/O operation is initiated for any of the data sets. For a description of associated data sets, see _OS and OS/VS Programming Support for the IBM 3505 Card Reader and IBM 3525 Card Punch_.

To support DEB validity checking, an OPEN macro instruction must be issued for every data extent block (DEB) created.

The standard form of the OPEN macro instruction is written as follows (the list and execute forms are shown following the description of the standard form):

[_symbol_]	OPEN	(_dcb address_,[(_options_)],...) ,[TYPE=J] ,[MODE=_24_|31]

dcb address—A-Type Address or (2-12)
The dcb address operand(s) specifies the address of the data control block(s) for the data set(s) to be prepared for processing.

options
The options operands shown in the following illustration indicate the volume positioning available based on the device type and access method being used. If option 1 is omitted, INPUT is assumed. If option 2 is omitted, DISP is assumed. Option 1 must be coded if option 2 is coded. Option 2 is ignored for SYSIN and SYSOUT data sets. Options 1 and 2 are ignored for BISAM and QISAM (in the scan mode), and the data control block indicates the operation. OUTPUT or OUTIN must be specified when creating a data set.

	DEVICE TYPE					
Access Method	**Magnetic Tape**		**Direct Access**		**Other Types**	
	Option 1	Option 2	Option 1	Option 2	Option 1	Option 2
QSAM	[INPUT] [EXTEND] [OUTPUT] [RDBACK]	[,REREAD] [,LEAVE] [,DISP]	[INPUT] [EXTEND] [OUTPUT] [UPDAT]	[,REREAD] [,LEAVE] [,DISP]	[INPUT] [EXTEND] [OUTPUT]	—
BSAM	[INPUT] [EXTEND] [OUTINX] [OUTPUT] [INOUT] [OUTIN] [RDBACK]	[,REREAD] [,LEAVE] [,DISP]	[INPUT] [EXTEND] [OUTINX] [OUTPUT] [INOUT] [OUTIN] [UPDAT]	[,REREAD] [,LEAVE] [,DISP]	[INPUT] [OUTPUT] [OUTPUT] [OUTPUT]	—
QISAM Load Mode	—	—	[OUTPUT] [EXTEND]	—	—	—
BPAM, BDAM	—	—	[INPUT] [OUTPUT] [UPDAT]	—	—	—

The following describes the options shown in the preceding illustration. All option operands are coded as shown.

Option 1	**Meaning**
EXTEND[1]	The data set is treated as an OUTPUT data set, except that records will be added to the end of the data set regardless of what was specified on the DISP parameter of the DD statement.
INPUT	Input data set.
INOUT[1]	The data set is first used for input and, without reopening, is used as an output data set. The data set is processed as INPUT if it is a SYSIN data set or if LABEL=(,,,IN) is specified in the DD statement.
OUTPUT	Output data set (for BDAM, OUTPUT is equivalent to UPDAT).
OUTIN[1]	The data set is first used for output and, without reopening, is used as an input data set. The data set is processed as OUTPUT if it is a SYSOUT data set or if LABEL=(,,,OUT) is specified in the DD statement.
OUTINX[1]	The data set is treated as an OUTIN data set, except that records will be added to the end of the data set regardless of what was specified on the DISP parameter of the DD statement.
RDBACK	Input data set, positioned to read backward. **Note:** Variable-length records cannot be read backward.
UPDAT	Data set to be updated in place or, for BDAM, blocks are to be updated or added.

[1] These options are not allowed for ISO/ANSI/FIPS Version 3 tape processing.

Option 2	Meaning
LEAVE	Positions the current tape volume to the logical end of the data set when volume switching occurs. If processing was forward, the volume is positioned to the end of the data set; if processing was backward (RDBACK), the volume is positioned to the beginning of the data set.
REREAD	Positions the current tape volume to reprocess the data set when volume switching occurs. If processing was forward, the volume is positioned to the beginning of the data set; if processing was backward (RDBACK), the volume is positioned to the end of the data set.
DISP	Specifies that a tape volume is to be disposed of in the manner implied by the DD statement associated with the data set. Direct access volume positioning and disposition are not affected by this parameter of the OPEN macro instruction. There are several dispositions that can be specified in the DISP parameter of the DD statement; DISP can be PASS, DELETE, KEEP, CATLG, or UNCATLG. This option has significance at the time an end-of-volume condition is encountered only when DISP is PASS. The end-of-volume condition may result from the issuance of an FEOV macro instruction or may be the result of reaching the end of a volume. If DISP is PASS in the DD statement, the tape will be spaced forward to the end of the data set on the current volume. If any DISP option is coded in the DD statement, (except when DISP is PASS), the resultant action at the time an end-of-volume condition arises depends on (1) how many tape units are allocated to the data set and (2) how many volumes are specified for the data set in the DD statement. This is determined by the UNIT and VOLUME parameters of the DD statement associated with the data set. If the number of volumes is greater than the number of units allocated, the current volume will be rewound and unloaded. If the number of volumes is less than or equal to the number of units, the current volume is merely rewound.

Note: When the DELETE option is specified, the system waits for the completion of the rewind operation before it continues processing subsequent reels of tape.

The **LEAVE** and **REREAD** options are meaningless except for magnetic tape and CLOSE **TYPE=T**. Any other options specified for CLOSE **TYPE=T** besides **LEAVE** and **REREAD** will be treated as **LEAVE** during execution.

TYPE=J

You can code OPEN **TYPE=J** to specify that, for each data control block referred to, you have supplied a job file control block (JFCB) to be used during initialization. A JFCB is an internal representation of information in a DD statement. This option, because it is used in conjunction with modifying a JFCB, should be used only by the system programmer or only under the system programmer's supervision. **MODE=31** is not allowed when **TYPE=J** is specified.

When you specify **TYPE=J**, you must also supply a DD statement. The amount of information in the DD statement is subject to discretion, but you must specify the device allocation and a ddname that corresponds to the associated data control block DCBDDNAM field.

For more detailed information on using **TYPE=J**, see System-Data Administration.

MODE=24|31

You can code OPEN **MODE=31** to specify a long form parameter list that will be able to contain 31-bit addresses. **MODE=31** is not permitted if **TYPE=J** is specified. You must be operating in 31-bit addressing mode in order to use the 31-bit addresses in the long form parameter list. The default, **MODE=24**, will specify a standard form parameter list with 24-bit addresses. If **TYPE=J** is specified, you must use the standard form parameter list.

The standard form parameter list is 4 bytes per entry. The standard form parameter list must reside below 16M, but the calling program may be above 16M. It is assumed that all ACBs and DCBs are below 16M.

The long form parameter list can reside above or below 16M. Each entry is 8 bytes long. Option information is contained in the first byte, zeros in the next three bytes, and the address of the ACB or DCB is contained in the last four bytes. Although the ACB or DCB address is contained in a 4-byte field, the DCB must be below 16M. Except for VSAM or VTAM ACBs, all ACBs must also be below 16M. Therefore, the leading byte of the ACB or DCB address must contain zeros. If the byte contains something other than zeros, an IEC190I message will be issued and the data set will not be opened. The program will not be abnormally terminated unless an attempt is made to read to or write from the data set.

Note: It is up to you to keep the mode specified in the **MF=L** and **MF=E** versions of the OPEN and CLOSE macros consistent. Errors and unpredictable results will occur if the specified modes are inconsistent.

Note: After the OPEN macro instruction has been executed, bit 3 of the DCBOFLGS field in the data control block is set to 1 if the data control block has been opened successfully, but is set to 0 if the data control block has not been opened successfully.

Note: To use the OPEN macro instruction supplied in the MVS/XA macro library on MVS/370, use the SPLEVEL macro instruction. You must use the SPLEVEL macro instruction to ensure that the MVS/XA version of the OPEN macro instruction executes successfully on MVS/370 DFP. For information on how to use the SPLEVEL macro, see System Macros and Facilities, Volume 2.

The following errors cause the results indicated:

Error	Result
Attempting to open a data control block that is already open.	No action.
Attempting to open a data control block when the dcb address operand does not specify the address of a data control block.	Unpredictable.
Attempting to open a DCB for a printer with the UCS feature and an error occurred when attempting to block or unblock data checks (specified by the presence or absence of OPTCD=U in the DCB macro).	Task abnormally terminated.
Attempting to open a data control block when a corresponding DD statement has not been provided.	A "DD STATEMENT MISSING" message is issued. An attempt to use the data set causes unpredictable results.

The last of these errors can be detected by testing bit 3 of the DCBOFLGS field in the data control block. Bit 3 is set to 0 in the case of an error and can be tested by the sequence:

```
    TM DCBOFLGS,X'10'

    BZ ERRORRTN       (Branch to user's error routine)
```

Executing the two instructions shown above requires writing a DCBD macro instruction in the program, and a base register must be defined with a USING statement before the instructions are executed.

OPEN—LIST FORM

The list form of the OPEN macro instruction is used to construct a data management parameter list. Any number of operands (data control block addresses and associated options) can be specified.

The list consists of a one-word entry for each DCB in the parameter list; the high-order byte is used for the options and the three low-order bytes are used for the DCB address. The end of the list is indicated by a 1 in the high-order bit of the last entry's option byte. The length of a list generated by a list form instruction must be equal to the maximum length list required by any execute form instruction that refers to the same list. A maximum length list can be constructed by one of two methods:

- Code a list-form instruction with the maximum number of parameters that are required by an execute form instruction that refers to the list.
- Code a maximum length list by using commas in a list-form instruction to acquire a list of the appropriate size. For example, coding OPEN (,,,,,,,,,),MF=L would provide a list of five fullwords (five dcb addresses and five options).

Entries at the end of the list that are not referenced by the execute-form instruction are assumed to have been filled in when the list was constructed or by a previous execute-form instruction. Before using the execute-form instruction, you may shorten the list by placing a 1 in the high-order bit of the last DCB entry to be processed.

A zeroed work area on a fullword boundary is equivalent to OPEN (,(INPUT,DISP),...),MF=L and can be used in place of a list-form instruction. The high-order bit of the last DCB entry must contain a 1 before this list can be used with the execute-form instruction.

A parameter list constructed by an OPEN, list-form, macro instruction can be referred to by either an OPEN or CLOSE execute form instruction.

The description of the standard form of the OPEN macro instruction provides the explanation of the function of each operand. The description of the standard form also indicates which operands are completely optional and those required in at least one of the pair of list and execute forms. The format description below indicates the optional and required operands in the list form only.

The list form of the OPEN macro is written:

[<u>symbol</u>]	**OPEN**	**([<u>dcb address</u>],[(<u>options</u>)],...)** **,MF=L** **[,TYPE=J]** **[,MODE=<u>24</u>|31]**

<u>dcb address</u>—A-Type Address

MF=L—Coded as shown
The **MF=L** operand specifies that the OPEN macro instruction is used to create a data management parameter list that is referenced by an execute form instruction.

TYPE=J
You can code OPEN **TYPE=J** to specify that, for each data control block referred to, you have supplied a job file control block (JFCB) to be used during initialization. A JFCB is an internal representation of information in a DD statement. This option, because it is used in conjunction with modifying a JFCB, should be used only by the system

programmer or only under the system programmer's supervision. **MODE=31** is not allowed when **TYPE=J** is specified.

When you specify **TYPE=J**, you must also supply a DD statement. The amount of information in the DD statement is subject to your discretion, but you must specify the device allocation and a ddname that corresponds to the associated data control block DCBDDNAM field.

For more detailed information on using **TYPE=J**, see System-Data Administration.

MODE=24|31
You can code OPEN **MODE=31** to specify a long form parameter list that will be able to contain 31-bit addresses. **MODE=31** is not permitted if **TYPE=J** is specified. You must be operating in 31-bit addressing mode in order to use the 31-bit addresses in the long form parameter list. The default, **MODE=24**, will specify a standard form parameter list with 24-bit addresses. If **TYPE=J** is specified, you must use the standard form parameter list.

The standard form parameter list is 4 bytes per entry. The standard form parameter list must reside below 16M, but the calling program may be above 16M. It is assumed that all ACBs and DCBs are below 16M.

The long form parameter list can reside above or below 16M. Each entry is 8 bytes long. Option information is contained in the first byte, zeros in the next three bytes, and the address of the ACB or DCB is contained in the last four bytes. Although the ACB or DCB address is contained in a 4-byte field, the DCB must be below 16M. Except for VSAM or VTAM ACBs, all ACBs must also be below 16M. Therefore, the leading byte of the ACB or DCB address must contain zeros. If the byte contains something other than zeros, an IEC190I message will be issued and the data set will not be opened. The program will not be abnormally terminated unless an attempt is made to read to or write from the data set.

Note: It is up to you to keep the mode specified in the **MF=L** and **MF=E** versions of the OPEN and CLOSE macros consistent. Errors and unpredictable results will occur if the specified modes are inconsistent.

OPEN—EXECUTE FORM

A remote data management parameter list is used in, and can be modified by, the execute form of the OPEN macro instruction. The parameter list can be generated by the list form of either an OPEN or CLOSE macro instruction.

The description of the standard form of the OPEN macro instruction provides the explanation of the function of each operand. The description of the standard form also indicates which operands are totally optional and those required in at least one of the pair of list and execute forms. The format description below indicates the optional and required operands in the execute form only.

The execute form of the OPEN macro is written:

[symbol]	OPEN	[([dcb address],[(options)],...)] ,MF=(E,{data management list address|(1)}) [,TYPE=J] [,MODE=24|31]

dcb address—RX-Type Address or (2-12)

MF=(E,{data management list address|(1)}

This operand specifies that the execute form of the OPEN macro instruction is used, and an existing data management parameter list (created by a list-form instruction) is used. The **MF=** operand is coded as follows:

E—Coded as shown

data management list address—RX-Type, (2-12), (1)

TYPE=J

You can code OPEN **TYPE=J** to specify that, for each data control block referred to, you have supplied a job file control block (JFCB) to be used during initialization. A JFCB is an internal representation of information in a DD statement. This option, because it is used in conjunction with modifying a JFCB, should be used only by the system programmer or only under the system programmer's supervision. **MODE=31** is not allowed when **TYPE=J** is specified.

When you specify **TYPE=J**, you must also supply a DD statement. The amount of information in the DD statement is subject to your discretion, but you must specify the device allocation and a ddname that corresponds to the associated data control block DCBDDNAM field.

For more detailed information on using **TYPE=J**, see System—Data Administration.

MODE=24|31

You can code OPEN **MODE=31** to specify a long form parameter list that will be able to contain 31-bit addresses. **MODE=31** is not permitted if **TYPE=J** is specified. You must be operating in 31-bit addressing mode in order to use the 31-bit addresses in the long form parameter list. The default, **MODE=24**, will specify a standard form parameter list with 24-bit addresses. If **TYPE=J** is specified, you must use the standard form parameter list.

The standard form parameter list is 4 bytes per entry. The standard form parameter list must reside below 16M, but the calling program may be above 16M. It is assumed that all ACBs and DCBs are below 16M.

The long form parameter list can reside above or below 16M. Each entry is 8 bytes long. Option information is contained in the first byte, zeros in the next three bytes, and the address of the ACB or DCB is contained in the last four bytes. Although the ACB or DCB address is contained in a 4-byte field, the DCB must be below 16M. Except for VSAM or VTAM ACBs, all ACBs must also be below 16M. Therefore, the leading byte of the ACB or DCB address must contain zeros. If the byte contains something other than zeros, an IEC190I message will be issued and the data set will not be opened. The program will not be abnormally terminated unless an attempt is made to read to or write from the data set.

Note: It is up to you to keep the mode specified in the **MF=L** and **MF=E** versions of the OPEN and CLOSE macros consistent. Errors and unpredictable results will occur if the specified modes are inconsistent.

PDAB—CONSTRUCT A PARALLEL DATA ACCESS BLOCK (QSAM)

The PDAB macro instruction is used in conjunction with the GET (**TYPE=P**) macro instruction. It defines an area in the problem program where the open and close routines build and maintain a queue of DCB addresses for use by the get routine.

The parallel data access block is constructed during the assembly of the problem program. The MAXDCB operand must be coded in the PDAB macro instruction, because it cannot be supplied from any other source.

Certain data set characteristics prevent a DCB address from being available on the queue—see the description of QSAM parallel input processing in Data Administration Guide.

The PDAB macro is written:

[symbol]	**PDAB**	**MAXDCB=**dcb number

MAXDCB=absexp (maximum value is 32767)
specifies the maximum number of DCBs that you require in the queue for a GET request.

Note: The number of bytes required for PDAB is equal to 24+8n, where n is the value of the keyword, MAXDCB.

PDABD—PROVIDE SYMBOLIC REFERENCE TO A PARALLEL DATA ACCESS BLOCK (QSAM)

The PDABD macro instruction is used to generate a dummy control section that provides symbolic names for the fields in one or more parallel data access blocks. The names, attributes, and descriptions of the fields appear in Appendix G, "PDABD Symbolic Field Names" on page 219.

The name of the dummy control section generated by a PDABD macro instruction is IHAPDAB. The use of any of the symbolic names provided by the dummy section should be preceded by a USING instruction specifying IHAPDAB and a dummy section base register containing the address of the actual parallel data access block. The PDABD macro instruction should only be used once within any assembled module; however, the resulting symbolic names can be used for any number of parallel data access blocks by changing the address in the dummy section base register. The PDABD macro instruction can be coded at any point in a control section. If coded at any point other than at the end of a control section, the control section must be resumed by coding a CSECT instruction.

The PDABD macro is written:

b	**PDABD**	b

POINT—POSITION TO A RELATIVE BLOCK (BPAM AND BSAM—TAPE AND DIRECT ACCESS ONLY)

The POINT macro starts the next READ or WRITE operation at the specified data set block on the current volume. Before you issue POINT macro, ensure that all input and output operations using the same data control block are tested for completion. If you are processing a data set that has been opened for UPDAT, you must issue a READ macro immediately after the POINT macro. If you are processing an output data set, you must issue a WRITE macro immediately after the POINT macro before you close the data set, unless you have already issued the CLOSE macro (with **TYPE=T** specified) before the POINT macro.

Note: If you specify the TYPE=T option in the CLOSE macro and you do not issue a WRITE macro before you close the data set, use the end-of-data location that is determined by TCLOSE.

The POINT macro is written:

[symbol]	**POINT**	dcb address ,block address ,[TYPE={ABS|REL}]

dcb address—RX-Type Address, (2-12), or (1)
The dcb address operand specifies the address of the data control block for the opened data set that is to be positioned.

block address—RX-Type Address, (2-12), or (0)
The block address operand indicates which block in the data set is to be processed next.

For an IBM 3480 Magnetic Tape subsystem, when TYPE=ABS is specified, the block address operand specifies the address of a fullword on a fullword boundary that contains the physical block identifier of the block in the data set that is to be processed next. This physical block identifier is provided as output from a prior execution of the NOTE macro.

When TYPE=REL is specified or defaults, the block address operand specifies the address of a fullword on a fullword boundary that contains the relative address of the block in the data set that is to be processed next. The relative address is specified as follows:

Magnetic Tape: The block number is in binary and is right-adjusted in the fullword with the high-order bits set to 0; add 1 if reading tape backward. Do not use the POINT macro instruction for tapes without standard labels when:

- The data set is opened for RDBACK, or
- The DD statement for the data set specifies DISP=MOD

If OPTCD=H is indicated in the data control block, the POINT macro instruction can be used to perform record positioning on DOS tapes that contain embedded checkpoint records. Any embedded checkpoint records that are encountered during the record positioning are bypassed and are not counted as blocks spaced over. OPTCD=H must be specified in a job control language DD statement. Do not use the POINT macro instruction to backspace DOS 7-track tapes that are written in data convert mode and that contain embedded checkpoint records.

Note: When an end-of-data condition is encountered on magnetic tape, you must not issue the POINT macro instruction unless you have first repositioned the tape for processing within your data set; otherwise, the POINT operation will be unsuccessful. (Issuing CLOSE TYPE=T is an easy method to use to accomplish repositioning in your EODAD routine.)

Direct Access Device: The fullword specified in the block address operand contains the relative track address (in the form TTRz), where:

TT	is a 2-byte relative track number.
R	is a 1-byte block (record) number on the track indicated by TT.
z	is a byte set to 0; it may also be set to 1 to retrieve the block following the TTR block.

Note: The first block of a magnetic tape data set is always specified by the hexadecimal value 00000001. The first block of a direct access device data set can be specified by either hexadecimal 00000001 or 00000100 (see the preceding description of TTRz).

TYPE={ABS|REL}
indicates whether the block address operand is a physical block identifier or a relative address.

ABS
indicates that the block address operand specifies an address of a fullword on a fullword boundary containing a physical block identifier of the block in the data set that is to be processed next.

REL
indicates that the block address operand specifies an address of a fullword on a fullword boundary containing the relative address of the block in the data set that is to be processed next.

POINT cannot be used for SYSIN or SYSOUT data sets.

If the volume cannot be positioned correctly or if the block identification is not of the correct format, the error analysis (SYNAD) routine is given control when the next CHECK macro instruction is executed.

COMPLETION CODES

When the system returns control to your problem program and you have specified the ABS parameter, the low-order byte of register 15 contains a return code; the low-order byte of register 0 contains a reason code:

Return Code (15)	Reason Code (0)	Meaning
00 (X'00')		Successful completion.
04 (X'04')		Device does not support block identifier.
08 (X'08')	01 (X'01')	Incorrect parameter.
08 (X'08')	02 (X'02')	Incorrect DCB or a DEBCHK error.
08 (X'08')	03 (X'03')	Environmental error.
08 (X'08')	11 (X'0B')	Unsuccessful call to ESTAE macro.
08 (X'08')	12 (X'0C')	Unsuccessful GETMAIN request.
12 (X'0C')		Input/output error.

PRTOV—TEST FOR PRINTER CARRIAGE OVERFLOW (BSAM AND QSAM—ONLINE PRINTER AND 3525 CARD PUNCH, PRINT FEATURE)

The PRTOV macro instruction is used to control the page format for an online printer when carriage control characters are not being used or to supplement the carriage control characters that are being used.

The PRTOV macro instruction causes the system to test for an overflow condition on the specified channel (either channel 9 or channel 12) of the printer carriage control, and either skip the printer carriage to the line corresponding to channel 1, or transfer control to the exit address, if one is specified. Overflow is detected after printing the line that follows the line corresponding to channel 9 or channel 12. The PRTOV macro should be issued each time you want the system to test for an overflow condition.

When the PRTOV macro instruction is used with a 3525 card punch, print feature, channel 9 or 12 can be tested. If an overflow condition occurs, control is passed to the overflow exit routine if the overflow exit address is coded, or a skip to channel 1 (first print-line of the next card) occurs.

When requesting overprinting (for example, to underscore a line), the PRTOV macro instruction is issued before the first PUT or WRITE macro instruction only. The PRTOV macro instruction should be issued only when the device type is an online printer. PRTOV cannot be used to request overprinting on the 3525. Overprinting cannot be performed on the 3800.

The PRTOV macro is written:

[symbol]	**PRTOV**	dcb address **,{9|12}** [,overflow exit address]

dcb address—RX-Type Address or (2-12)
: The dcb address operand specifies the address of the data control block opened for output to an online printer or 3525 card punch with a print feature.

9—Coded as shown
12—Coded as shown
: These operands specify the channel that is to be tested by the PRTOV macro instruction. For an online printer, 9 and 12 correspond to carriage control channels 9 and 12. For the 3525 card punch, 9 corresponds to print line number 17, and 12 corresponds to print line number 23. More detail about the card print-line format is included in *OS and OS/VS Programming Support for the IBM 3505 Card Reader and IBM 3525 Card Punch*.

overflow exit address—RX-Type Address or (2-12)
: The overflow exit address operand specifies the address of the user-supplied routine to be given control when an overflow condition is detected on the specified channel. If this operand is omitted, the printer carriage skips to the first line of the next page or the 3525 skips to the first line of the next card before executing the next PUT or WRITE macro instruction.

When the overflow exit routine is given control, the contents of the registers are as follows:

Register	Contents
0 and 1	The contents are destroyed.
2 - 13	The same contents as before the macro instruction was executed.
14	Return address.
15	Overflow exit routine address.

PUT—WRITE NEXT LOGICAL RECORD (QISAM)

The PUT macro instruction causes the system to write a record into an indexed sequential data set. If the move mode is used, the PUT macro instruction moves a logical record into an output buffer from which it is written. If locate mode is specified, the address of the next available output buffer segment is available in register 1 after the PUT macro instruction is executed. The logical record can then be constructed in the buffer for output as the next record. The records are blocked by the system (if specified in the data control block) before being placed in the data set. The system uses the length specified in the record length (DCBLRECL) field of the data control block as the length of the record currently being written. When constructing blocked variable-length records in the locate mode, the problem program may either specify the maximum record length once in the DCBLRECL field of the data control block or provide the actual record length in the DCBLRECL field before issuing each PUT macro instruction. Use of the maximum record length may result in more but shorter blocks, because the system uses this length when it tests to see if the next record can be contained in the current block.

The PUT macro instruction is used to create or extend an indexed sequential data set. To extend the data set, the key of any added record must be higher than the highest key existing in the data set, and the disposition parameter of the DD card must be specified as DISP=MOD. The new records are placed in the prime data space, starting in the first available space, until the original space allocation is exhausted.

To create a data set using previously allocated space, the disposition parameter of the DD card must specify DISP=OLD.

The PUT macro is written:

[<u>symbol</u>]	**PUT**	<u>dcb address</u> [,<u>area address</u>]

<u>dcb address</u>—RX-Type Address, (2-12), or (1)
The dcb address operand specifies the address of the data control block for the opened ISAM data set.

<u>area address</u>—RX-Type Address, (2-12), or (0)
The area address operand specifies the address of the area that contains the record to be written (move mode only). Either move or locate mode can be used with QISAM, but they must not be mixed within the specified data control block. The following describes operations for locate and move modes:

Locate Mode: If locate mode is specified in the data control block, the area address operand must be omitted. The system returns the address of the next available buffer in register 1; this is the buffer into which the next record is placed. The record is not written until another PUT macro instruction is issued for the same data control block. The last record is written when a CLOSE macro instruction is issued to close the data set.

Move Mode: If move mode has been specified in the data control block, the area address operand must specify the address in the problem program that contains the record to be written. The system moves the record from the area to an output buffer before control is returned. If the area address operand is omitted, the system assumes that register 0 contains the area address.

PUT ROUTINE EXIT

The error analysis (SYNAD) routine is given control if the output operation could not be completed satisfactorily. The contents of the registers when the error analysis routine is given control are described in Appendix A, "Status Information Following an Input/Output Operation" on page 192.

PUT—WRITE NEXT LOGICAL RECORD (QSAM)

The PUT macro instruction causes the system to write a record in a sequential data set. Various modes are available and are specified in the DCB macro instruction. In the locate mode, the address of an area within an output buffer is returned in register 1 after the macro instruction is executed. The user should subsequently construct, at this address, the next sequential record or record segment. The move mode of the PUT macro instruction causes a logical record to be moved into an output buffer. In the data mode, which is available only for variable-length spanned records, the PUT macro instruction moves only the data portion of the record into one or more output buffers.

The records are blocked by the control program (as specified in the data control block) before being placed in the data set. For undefined-length records, the DCBLRECL field determines the length of the record that is subsequently written. For variable-length records, the DCBLRECL field is used to locate a buffer segment of sufficient size (locate mode), but the length of the record actually constructed is verified before the record is written (the output block can be filled to the maximum if, before issuing the PUT macro, DCBLRECL is set equal to the record length). For variable-length spanned records, the system segments the record according to the record length, buffer length, and amount of unused space remaining in the output buffer. The smallest segment created will be 5 bytes, 4 for the segment descriptor word plus 1 byte of data.

If the ISCII/ASCII translation routines are included when the operating system is generated, translation can be requested by coding LABEL=(,AL) or (,AUL) in the DD statement, or it can be requested by coding OPTCD=Q in the DCB macro instruction or DCB subparameter of the DD statement. When translation is requested, all QSAM records whose record format (RECFM operand) is F, FB, D, DS, DB, DBS, or U are automatically translated from EBCDIC code to ISCII/ASCII code. For translation to occur correctly, all output data must be in EBCDIC code; any EBCDIC character that cannot be translated into an ISCII/ASCII character is replaced by a substitute character.

The PUT macro is written:

[<u>symbol</u>]	**PUT**	<u>dcb address</u> [,<u>area address</u>]

<u>dcb address</u>—RX-Type Address, (2-12), or (1)
The dcb address operand specifies the address of the data control block for the data set opened for output.

<u>area address</u>—RX-Type Address, (2-12), or (0)
The area address operand specifies the address of an area that contains the record to be written (move or data mode). The move, locate, or data mode can be used with QSAM, but they must not be mixed within the specified data control block. If the area address operand is omitted in the move or data mode, the system assumes that register zero contains the area address. The following describes the operation of the three modes:

Locate Mode: If locate mode is specified, the area address operand must be omitted. The system returns the address of the next available buffer in register 1; this is the buffer into which the next record is placed.

When variable-length spanned records are processed without the extended logical record interface (XLRI), and a record area has been provided for a logical record interface (LRI) (BFTEK=A has been specified in the data control block or a BUILDRCD macro instruction has been issued), the address returned in register 1 points to an area large enough to contain the maximum record size (up to 32756 bytes). The system segments the record and writes all segments, providing proper control codes for each segment. If, for variable-length spanned records, an area has not been provided, the actual length remaining in the buffer will be returned in register 0. In this case, it is the user's responsibility to segment the records and process them in terms of record segments. ISO/ANSI/FIPS spanned records, RECFM=DS or RECFM=DBS, may not be processed in segment mode. The record or segment is not written until another PUT macro instruction is issued for the same data control block. The last record is written when the CLOSE macro instruction is issued.

When a PUT macro instruction is used in the locate mode, the address of the buffer for the first record or segment is obtained by issuing a PUT macro after open. QSAM returns the address in register 1. The user then moves data to this address. The buffer is not written to the data set until the next PUT macro is issued. If records are blocked, the data is not written to the data set until the PUT following the one that filled the buffer. Each PUT macro will return the address of the next buffer in register 1. After this address is given to the user, QSAM will always count this address as a valid record. The user should always place valid data at the address returned in register 1 before issuing another PUT or FEOV or CLOSE MACRO; otherwise, residual data at that location will be written to the data set. After an FEOV macro is issued, (for multivolume data sets), register 1 must be reinitialized with the first buffer address for the next volume by issuing a PUT macro after return from FEOV.

Move Mode: If move mode has been specified in the data control block, the area address operand specifies the address of the area that contains the record to be written. The system moves the record to an output buffer before control is returned.

Data Mode: If data mode is specified in the data control block (data mode can be specified for variable-length spanned records only), the area address operand specifies the address of an area in the problem program that contains the data portion of the record to be written. The system moves the data portion of the record to an output buffer before control is returned. The user must place the total data length in the DCBPRECL (not the DCBLRECL) field of the data control block before the PUT macro instruction is issued.

Extended Logical Record Interface (XLRI): When the PUT macro is used with the extended logical record interface, the address returned in register 1 points to an area that is used to build a 4-byte logical record length field (RDW) followed by a complete logical record. The logical record length byte count occupies the three low-order bytes of the record length field and must include the length of the field. The high-order byte must be zero. The DCB LRECL value indicates the length of the longest logical record of the data set in 'K' (1024-byte) units.

PUT ROUTINE EXIT

If the output operation could not be completed satisfactorily, the error analysis (SYNAD) routine is given control after the next PUT instruction is issued. The contents of the registers when the error analysis routine is given control are described in Appendix A, "Status Information Following an Input/Output Operation" on page 192.

PUTX—WRITE A RECORD FROM AN EXISTING DATA SET (QISAM AND QSAM)

The PUTX macro instruction causes the control program to return an updated record to a data set (QISAM and QSAM) or to write a record from an input data set into an output data set (QSAM only). There are two modes of the PUTX macro instruction. The output mode (QSAM only) allows writing a record from an input data set on a different output data set. The output data set may specify the spanning of variable-length records, but the input data set must not contain spanned records.

The update mode returns an updated record to the data set from which it was read. The logical records are blocked by the control program, as specified in the data control block, before they are placed in the output data set. The control program uses the length specified in the DCBLRECL field as the length of the record currently being stored. Control is not returned to the user's program until the control program has processed the record.

For SYSIN or SYSOUT data sets, the PUTX macro instruction can be used only in the output mode.

The PUTX macro is written:

[symbol]	**PUTX**	dcb address [,input dcb address]

dcb address—RX-Type Address, (2-12), or (1)
The dcb address operand specifies the address of the data control block for a data set opened for output.

input dcb address—RX-Type Address, (2-12), or (0)
The input dcb address operand specifies the address of a data control block opened for input. The PUTX macro instruction can be used for the following modes:

Output Mode: This mode is used with QSAM only. The input dcb address operand specifies the address of the data control block opened for input. If this operand is omitted, the system assumes that register 0 contains the input dcb address.

Update Mode: The input dcb address operand is omitted for update mode.

PUTX ROUTINE EXIT

The error analysis (SYNAD) routine is given control if the operation is not completed satisfactorily. The contents of the registers when the error analysis routine is given control are described in Appendix A, "Status Information Following an Input/Output Operation" on page 192.

READ—READ A BLOCK (BDAM)

The READ macro instruction causes a block to be retrieved from a data set and placed in a designated area of storage. Control may be returned to the problem program before the block is retrieved. The input operation must be tested for completion using a CHECK or WAIT macro instruction. A data event control block, shown in Appendix A, "Status Information Following an Input/Output Operation" on page 192, is constructed as part of the macro expansion.

The standard form of the READ macro instruction is written as follows (the list and execute forms are shown following the descriptions of the standard form):

[symbol]	**READ**	decb name ,type ,dcb address ,{area address|'S'} ,{length|'S'} ,{key address|'S'|0} ,block address [,next address]

decb name—symbol
The decb name operand specifies the name assigned to the data event control block created as part of the macro expansion.

type—{DI[F|X][R|RU]}
{DK[F|X][R|RU]}
The type operand is coded in one of the combinations shown above to specify the type of read operation and the optional services performed by the system:

DI
specifies that the data and key, if any, are to be read from a specific device address. The device address, which can be designated by any of the three addressing methods, is supplied by the block address operand.

DK
specifies that the data (only) is to be read from a device address identified by a specific key. The key to be used as a search argument must be supplied in the area specified by the key address operand; the search for the key starts at the device address supplied in the area specified by the block address operand. The description of the DCB macro instruction, LIMCT operand, contains a description of the search.

F
requests that the system provide block position feedback into the area specified by the block address operand. This character can be coded as a suffix to **DI** or **DK** as shown above.

X
requests exclusive control of the data block being read, and that the system provide block position feedback into the area specified by the block address operand. The descriptions of the WRITE and RELEX macro instructions contain a description of releasing a data block that is under exclusive control. This character can be coded as a suffix to **DI** or **DK** as shown above.

R

requests that the system provide next address feedback into the area specified by the next address operand. When R is coded, the feedback is the relative track address of the next data record. This character can be coded as a suffix to **DI** or **DK, DIF, DIX, DKF**, or **DKX** as shown above, but can be coded only for use with variable-length spanned records.

RU

requests that the system provide next address feedback into the area specified by the next address operand. When **RU** is coded, the feedback is the relative track address of the next capacity record (R0) or data record whichever occurs first. These characters can be coded as a suffix to **DI, DK, DIF, DIX, DKF**, or **DKX**, but it can be coded only for use with variable-length spanned records.

dcb address—A-Type Address or (2-12)
The dcb address operand specifies the address of the data control block opened for the data set to be read.

area address—A-Type Address, (2-12), or **'S'**
The area address operand specifies the address of the area in which the data block is to be placed. If **'S'** is coded instead of an address, dynamic buffering is requested (dynamic buffering must also be specified in the MACRF operand of the DCB macro instruction). When dynamic buffering is used, the system acquires a buffer and places its address in the data event control block.

length—symbol, decimal digit, absexp, (2-12), or **'S'**
The length operand specifies the number of data bytes to be read up to a maximum of 32760. If **'S'** is coded instead of a length, the number of bytes to be read is taken from the data control block. This operand is ignored if the records are not format-U.

key address—A-Type Address, (2-12), **'S'**, or **0**
The key address operand specifies the address of the area for the key of the desired data block. If the search operation is made using a key, the area must contain the key. Otherwise, the key is read into the designated area. If the key is read and **'S'** was coded for the area address, **'S'** can also be coded for the key address; the key and data are read sequentially into the buffer acquired by the system. If the key is not to be read, specify **0** instead of an address or **'S'**.

block address—A-Type Address or (2-12)
The block address operand specifies the address of the area containing the relative block address, relative track address, or actual device address of the data block to be retrieved. The device address of the data block retrieved is placed in this area if block position feedback is requested. The length of the area that contains the address depends on whether the feedback option (OPTCD=F) has been specified in the data control block and if the READ macro instruction requested feedback.

If OPTCD=F has been specified, feedback (if requested) is in the same form as originally presented by the READ macro instruction, and the field can be either 3 or 8 bytes long, depending on the type of addressing.

If OPTCD=F has not been specified, feedback (if requested) is in the form of an actual device address, and the field must be 8 bytes long.

next address—A-Type Address or (2-12)
The next address operand specifies the address of the storage area in which the system places the relative address of the next record. The length operand must be specified as **'S'**. When the next address operand is specified, an **R** or **RU** must be added to the type operand (for example, **DIR** or **DIRU**). The R indicates that the next address returned is the next data record. **RU** indicates that the next address returned is for the next data or capacity record, whichever occurs first. The next address operand can be coded only for use with variable-length spanned records.

READ—READ A BLOCK OF RECORDS (BISAM)

The READ macro instruction causes an unblocked record, or a block containing a specified logical record, to be retrieved from a data set. The block is placed in a designated area of storage, and the address of the logical record is placed in the data event control block. The data event control block is constructed as part of the macro expansion and is described in Appendix A, "Status Information Following an Input/Output Operation" on page 192.

Control may be returned to the problem program before the block is retrieved. The input operation must be tested for completion using a WAIT or CHECK macro instruction.

The standard form of the READ macro instruction is written as follows for BISAM (the list and execute forms are shown following the descriptions of the standard form):

[symbol]	**READ**	decb name ,type ,dcb address ,{area address|'S'} ,{length|'S'} ,key address

decb name—symbol
: The decb name operand specifies the name assigned to the data event control block (DECB) created as part of the macro expansion.

type—{K|KU}
: The type operand is coded as shown to specify the type of read operation:

K
: specifies normal retrieval.

KU
: specifies that the record retrieved is to be updated and returned to the data set; the system saves the device address to be returned.

When an ISAM data set is being updated with a READ KU macro instruction and a WRITE K macro instruction, both the READ and WRITE macro instructions must reference the same data event control block. This update operation can be performed by using a list-form instruction to create the list (data event control block) and by using the execute form of the READ and WRITE macro instructions to reference the same list.

dcb address—A-Type Address or (2-12)
: The dcb address operand specifies the address of the data control block for the opened data set to be read.

area address—A-Type Address, (2-12), or 'S'
: The area address operand specifies the address of the area in which the data block is placed. The first 16 bytes of this area are used by the system and do not contain information from the data block. The area address must specify a different area than the key address. Dynamic buffering is specified by coding 'S' instead of an address; the address of the acquired storage area is returned in the data event control block. Indexed sequential buffer and work area requirements are described in *Data Administration Guide*.

<u>length</u>—symbol, decimal digit, absexp, (2-12), or 'S'
The length operand specifies the number of bytes to be read up to a maximum of 32760. If 'S' is coded instead of a length, the number of bytes to be read is taken from the count field of the record; for blocked records, 'S' <u>must</u> be coded.

<u>key address</u>—A-Type Address or (2-12)
The key address operand specifies the address of the area in the problem program containing the key of a logical record in the block that is to be retrieved. When the input operation is completed, the storage address of the logical record is placed in the data event control block. The key address must specify a different area than the area address.

READ—READ A BLOCK (BPAM AND BSAM)

The READ macro instruction causes a block to be retrieved from a data set and placed in a designated area of storage. Control may be returned to the problem program before the block is retrieved. The input operation must be tested for completion using a CHECK macro instruction. A data event control block, shown in Appendix A, "Status Information Following an Input/Output Operation" on page 192, is constructed as part of the macro expansion.

If the OPEN macro instruction specifies UPDAT, both the READ and WRITE macro instructions must reference the same data event control block. (See the list form of the READ or WRITE macro instruction for a description of how to construct a data event control block; see the execute form of the READ or WRITE macro instruction for a description of how to modify an existing data event control block.)

The standard form of the READ macro instruction is written as follows (the list and execute forms are shown following the descriptions of the standard form instructions):

[symbol]	**READ**	decb name ,type ,dcb address ,area address [,length|,'S']

decb name—symbol
    The decb name operand specifies the name assigned to the data event control block (DECB) created as part of the macro expansion.

type—{SF|SB}
    The type operand is coded as shown to specify the type of read operation:

SF
    specifies normal, sequential, forward retrieval.

SB
    specifies a read-backward operation; this operand can be specified only for magnetic tape with format-F or format-U records.

    This operand is intended to be used when the data set is OPEN for RDBACK. Tape positioning, label processing, and volume mounting errors will occur during EOV and CLOSE if an OPEN option, other than RDBACK, is used.

dcb address—A-Type Address or (2-12)
    The dcb address operand specifies the address of the data control block for the opened data set to be read.

area address—A-Type Address or (2-12)
    The area address operand specifies the address of the problem program area in which the record is placed. When a READ SB macro instruction is issued, the area address must be the address of the last byte of the area into which the record is read. If the data set contains keys, the key is read into the buffer followed by the data.

length—symbol, decimal digit, absexp, (2-12), or 'S'
    The length operand specifies the number of data bytes to be read, to a maximum of 32760. If the data is translated from ISCII/ASCII code to EBCDIC code, the maximum number of bytes that can be read is 2048. For format-U records, 'S' or a valid length must be coded. The number of bytes to be read is taken from the data control block if 'S' is coded instead of a number. (This operand is ignored for format-F

or format-V records.) For format-D records only, the length of the record just read is automatically inserted into the DCBLRECL field by the check routine if BUFOFF=(L) is not specified in the data control block.

READ—READ A BLOCK (OFFSET READ OF KEYED BDAM DATA SET USING BSAM)

The READ macro instruction causes a block to be retrieved from a data set and placed in a designated area of storage. The data set is a BDAM data set and its record format is unblocked variable-length spanned records. BFTEK=R must be specified in the data control block. Control may be returned to the problem program before the block is retrieved. The input operation must be tested for completion using a CHECK macro instruction. A data event control block, shown in Appendix A, "Status Information Following an Input/Output Operation" on page 192, is constructed as part of the macro expansion.

The standard form of the READ macro instruction is written as follows (the list and execute forms are shown following the descriptions of the standard form):

[<u>symbol</u>]	**READ**	<u>decb name</u> **,SF** ,<u>dcb address</u> ,<u>area address</u>

<u>decb name</u>—symbol
: The decb name operand specifies the name assigned to the data event control block (DECB) created as part of the macro expansion.

SF
: specifies normal, sequential, forward retrieval.

<u>dcb address</u>—A-Type Address or (2-12)
: The dcb address operand specifies the address of the data control block for the opened BDAM data set to be read.

<u>area address</u>—A-Type Address or (2-12)
: The area address operand specifies the address of the area in which the record is placed.

When a spanned BDAM data set is created with keys, only the first segment of a record has a key; successive segments do not. When a spanned record is retrieved by the READ macro instruction, the system places a segment in a designated area addressed by the area address operand. The problem program must assemble all the segments into a logical record. Because only the first segment has a key, the successive segments are read into the designated area offset by key length to ensure that the block-descriptor word and the segment-descriptor word are always in their same relative positions.

READ—LIST FORM

The list form of the READ macro instruction is used to construct a data management parameter list in the form of a data event control block (DECB). For a description of the various fields of the DECB for each access method, see Appendix A, "Status Information Following an Input/Output Operation" on page 192.

The description of the standard form of the READ macro instruction provides the explanation of the function of each operand. The description of the standard form also indicates the operands used for each access method, and the meaning of 'S' when coded for the area address, length, and key address operands. For each access method, 'S' can be coded only for those operands for which it can be coded in the standard form of the macro instruction. The format description below indicates the optional and required operands in the list form only.

The list form of the READ macro is written:

[symbol]	READ	decb name ,type ,[dcb address] ,[area address|'S'] ,[length|'S'] ,[key address|'S'] ,[block address] ,[next address] ,MF=L

decb name—symbol

type—Code one of the types shown in the standard form

dcb address—A-Type Address

area address—A-Type Address or 'S'

length—symbol, decimal digit, absexp, or 'S'

key address—A-Type Address or 'S'

block address—A-Type Address

next address—A-Type Address

MF=L—Coded as shown
The MF=L operand specifies that the READ macro instruction is used to create a data event control block that can be referenced by an execute-form instruction.

READ—EXECUTE FORM

A remote data management parameter list (data event control block) is used in, and can be modified by, the execute form of the READ macro instruction. The data event control block can be generated by the list form of either a READ or WRITE macro instruction.

The description of the standard form of the READ macro instruction provides the explanation of the function of each operand. The description of the standard form also indicates the operands used for each access method and the meaning of 'S' when coded for the area address, length, and key address operands. For each access method, 'S' can be coded only for those operands for which it can be coded in the standard form of the macro instruction. The format description below indicates the optional and required operands in the execute form only.

The execute form of the READ macro is written:

[symbol]	READ	decb address ,type ,[dcb address] ,[area address|'S'] ,[length|'S'] ,[key address|'S'] ,[block address] ,[next address] ,MF=E

decb address—RX-Type Address or (2-12)

type—Code one of the types shown in the standard form

dcb address—RX-Type Address or (2-12)

area address—RX-Type Address, (2-12), or 'S'

length—symbol, decimal digit, absexp, (2-12), or 'S'

key address—RX-Type Address, (2-12), or 'S'

block address—RX-Type Address, or (2-12)

next address—RX-Type Address or (2-12)

MF=E—Coded as shown
The **MF=E** operand specifies that the execute form of the READ macro instruction is used, and that an existing data event control block (specified in the decb address operand) is used by the access method.

RELEX—RELEASE EXCLUSIVE CONTROL (BDAM)

The RELEX macro instruction causes release of a data block from exclusive control. The block must have been requested in an earlier READ macro instruction that specified either **DIX** or **DKX**.

Note: A WRITE macro instruction that specifies either **DIX** or **DKX** can also be used to release exclusive control.

The RELEX macro is written:

[symbol]	RELEX	**D** ,dcb address ,block address

D
specifies direct access.

dcb address—RX-Type Address, (2-12), or (1)
The dcb address operand specifies the address of the data control block for a BDAM data set opened for processing. The operand must specify the same data control block designated in the associated READ macro instruction.

block address—RX-Type Address, (2-12), or (0)
The block address operand specifies the address of the area containing the relative block address, relative track address, or actual device address of the data block being released. The operand must specify the same area designated in the block address operand of the associated READ macro instruction.

COMPLETION CODES

When the system returns control to the problem program, the low-order byte of register 15 contains one of the following return codes; the three high-order bytes of register 15 are set to 0.

Return Code (15)	Meaning
00 (X'00')	Operation completed successfully.
04 (X'04')	The specified data block was not in the exclusive control list.
08 (X'08')	The relative track address, relative block address, or actual device address was not within the data set.

RELSE—RELEASE AN INPUT BUFFER (QISAM AND QSAM INPUT)

The RELSE macro instruction causes immediate release of the current input buffer. The next GET macro instruction retrieves the first record from the next input buffer. For variable-length spanned records (QSAM), the input data set is spaced to the next segment that starts a logical record in a subsequent block. Thus, one or more blocks of data or records may be skipped. The RELSE macro instruction is ignored if a buffer has just been completed or released, if the records are unblocked, or if issued for a SYSIN data set.

The RELSE macro is written:

[symbol]	**RELSE**	dcb address

dcb address—RX-Type Address, (2-12), or (1)
The dcb address operand specifies the address of the data control block for the opened input data set.

SETL—SET LOWER LIMIT OF SEQUENTIAL RETRIEVAL (QISAM INPUT)

The SETL macro instruction causes the control program to start processing the next input request at the specified record or device address. Sequential retrieval of records using the GET macro instruction continues from that point until the end of the data set is encountered or a CLOSE or ESETL macro instruction is issued. An ESETL macro instruction must be issued between SETL macro instructions that specify the same data set.

The SETL macro instruction can specify that retrieval is to start at the beginning of the data set, at a specific address on the device, at a specific record, or at the first record of a specific class of records. For additional information on SETL functions, see Data Administration Guide.

The SETL macro is written:

[symbol]	**SETL**	dcb address
		{,K[H],lower limit address}
		{,KC,lower limit address}
		{,KD[H],lower limit address}
		{,KCD,lower limit address}
		{,I,lower limit address}
		{,ID,lower limit address}
		{,B}
		{,BD}

dcb address—RX-Type Address, (2-12), or (1)
The dcb address operand specifies the address of the data control block opened for the indexed sequential data set being processed.

The following operands are coded as shown; they specify the starting point and type of retrieval:

K
specifies that the next input operation is to begin at the record containing the key specified in the lower-limit address operand.

KC
specifies that the next input operation is to begin at the first record of the key class specified in the lower-limit address operand. If the first record of the specified key class has been deleted, retrieval begins at the next nondeleted record regardless of key class.

H
used with either K or KD, specifies that, if the key in the lower-limit address operand is not in the data set, retrieval begins at the next higher key. The character **H** cannot be coded with the key class operands (**KC** and **KCD**).

KD
specifies that the next input operation is to begin at the record containing the key specified in the lower-limit address operand, but only the data portion of the record is retrieved. This operand is valid only for unblocked records.

KCD
specifies that the next input operation is to begin at the first record of the key class specified in the lower-limit address operand, but only the data portion of the record is retrieved. This operand is valid only for unblocked records.

I
specifies that the next input operation is to begin with the record at the actual device address specified in the lower-limit address operand.

ID
specifies that the next input operation is to begin with the record at the actual device address specified in the lower-limit address operand, but only the data portion of the record is retrieved. This operand is valid only for unblocked records.

B
specifies that the next input operation is to begin with the first record in the data set.

BD
specifies that the next input operation is to begin with the first record in the data set, but only the data portion is retrieved. This operand is valid only for unblocked records.

<u>lower limit address</u>—RX-Type Address, (2-12), or (0)
The lower-limit address operand specifies the address of the area containing the key, key class, or actual device address that designates the starting point for the next input operation. If I or ID has been specified, this area must contain the actual device address (in the form MBBCCHHR) of a prime data record; the other types require that the key or key class be contained in this area.

SETL EXIT

The error analysis (SYNAD) routine is given control if the operation could not be completed successfully. The exception condition code and general registers are set as shown in Appendix A, "Status Information Following an Input/Output Operation" on page 192. If the SETL macro instruction is not reissued, retrieval starts at the beginning of the data set.

SETPRT—PRINTER SETUP (BSAM, QSAM, AND EXCP)

For the IBM 3800 Printing Subsystem, the SETPRT macro instruction is used to initially set or dynamically change the printer control information. The following control information may be changed with the SETPRT macro:

- Bursting of forms (BURST parameter)
- Character arrangements to be used (CHARS parameter)
- The number of copies (COPIES parameter)
- The starting copy number (COPYNR parameter)
- Vertical formatting of a page (FCB parameter)
- Flashing of forms (FLASH parameter)
- Initializing the printer control information (INIT parameter)
- Modification of copy (MODIFY parameter)
- Blocking or unblocking of data checks (OPTCD parameter)

In addition to changing the control information, you can also do the following:

- Create your own 3800 load modules in a library to replace the use of SYS1.IMAGELIB (LIBDCB parameter).
- SETPRT error messages that are sent to the printer can also be passed back to the invoking program (MSGAREA parameter).
- Print or suppress error messages on the printer (PRTMSG parameter).
- Control the scheduling of SYSOUT segment printing (DISP parameter).

To use all-points addressability when operating the 3800 Model 3, PSF libraries (for example, SYS1.FONTLIB, SYS1.FDEFLIB, SYS1.PDEFLIB) will be used instead of SYS1.IMAGELIB. As an alternative, the library with the LIBDCB parameter may be used.

For additional information on how to use the SETPRT macro instruction with the 3800 Model 3, see _IBM 3800 Printing Subsystem Models 3 and 8 Programmer's Guide_.

For printers other than the 3800 Printing Subsystem, the SETPRT macro instruction is used to control the following:

- Selection and verification of UCS and FCB images (UCS and FCB parameters)
- Blocking or unblocking of data checks (OPTCD parameter)
- Printing lowercase EBCDIC characters in uppercase (OPTCD and UCS parameters)

For printers that have a universal character set (UCS) buffer or a forms control buffer (FCB), the SETPRT macro instruction is used to fetch UCS and FCB images from the image library (SYS1.IMAGELIB) and load them into their respective buffers. Note that FCB images for the 3203/3211 and 3800 are not compatible. The universal character sets for the 1403/3203 and the character arrangement table modules for the 3800 are also not compatible.

IBM-supplied UCS images, FCB images, and character arrangement table modules are included in SYS1.IMAGELIB at system generation time. For impact printers, user-defined character sets and forms control images can be added to SYS1.IMAGELIB as described in System-Data Administration. For the 3800, user-defined character arrangement table modules, copy modification modules, FCB modules, and graphic character modification modules (that modify the character set) and library character sets can be added to SYS1.IMAGELIB as described in Utilities. The EXLST parameter of the DCB macro instruction can be used to specify the address of an FCB module in storage.

For the 1403 and 3211, if the specified UCS or FCB image cannot be found in SYS1.IMAGELIB or the DCB exit list, the system operator is asked to specify a replacement name and can therefore override an error made by the program. For the 3800, the SETPRT routines never ask the operator to replace or respecify a parameter from the SETPRT macro, and SETPRT processing is terminated.

When BSAM is being used, all write operations must be checked for completion before the SETPRT macro instruction is issued; any incomplete write operations are purged. Issuing the SETPRT macro instruction for a device other than an on-line UCS printer or the 3800 Printing Subsystem results in an error return code.

The standard form of the SETPRT macro instruction is written as follows (the list and execute forms are shown following the standard form):

```
[symbol]  SETPRT  dcbaddr
                  [,BURST={N|Y}]
                  [,CHARS={name|A(address)|R(register)}}
                          {({name|A(address)|
                          R(register)},...)}]
                  [,COPIES=number]
                  [,COPYNR=number]
                  [,DISP=[SCHEDULE|NOSCHEDULE|EXTERNAL]
                  [,FCB={imageid|A(address)|R(register)}
                        {({imageid|A(address)|
                        R(register)},{V|A})}]
                  [,FLASH={name}
                          {([name],count)}]
                  [,INIT={N|Y}]
                  [,LIBDCB=dcbaddress]
                  [,MODIFY={{name|A(address)|R(register)}
                           {({name|A(address)|
                           R(register)},trc)}]
                  [,MSGAREA=address]
                  [,OPTCD={B|U}
                          {({B|U},{F|U})}]
                  [,PRTMSG=[N|Y]
                  [,REXMIT={N|Y}]
                  [,UCS={csc}
                        {(csc,{F|F,V|V})}]
```

dcbaddr—A-Type Address or (2-12)
    The dcbaddr operand specifies the address of the data control block for the data set to be printed; the data set must be opened for output before the SETPRT macro instruction is issued.

BURST={N|Y}
    The BURST operand specifies whether the paper output is to be burst. BURST=Y indicates that the printed output is to be burst into separate sheets and stacked. BURST=N indicates that the printed output is to go into the continuous forms stacker. If BURST is not specified, the SETPRT routine assumes BURST=N. If bursting is requested, the printed output is threaded into the burster-trimmer-stacker. Otherwise, the printed output is threaded into the continuous forms stacker. The operand causes a message to be printed at the system console telling the operator to thread the paper again if needed. This operand is valid for the 3800 printer only.

CHARS={name|A(address|R(register)**}**
{({name|A(address)|R(register)**},...)}**
The CHARS operand specifies one to four character arrangement tables to be used when printing a data set. This operand is valid for the 3800 printer only.

name
: is the last four characters of the 8-byte member name for a character arrangement table module. For information on the modules available, see IBM 3800 Printing Subsystem Programmer's Guide.

A(address)
: specifies an in-storage address of the user-provided character arrangement table module. For information on the format of the module, see Utilities.

R(register)
: specifies the register that contains an in-storage address of the user-provided character arrangement table module. For information on the format of the module, see Utilities.

COPIES=number
specifies the total number of copies of each page of the data set that is to be printed (from 1 to 255) before going to the next page. If the COPIES operand is omitted, one copy of each page is printed. This operand is valid for the 3800 printer only.

DISP={SCHEDULE|NOSCHEDULE|EXTERNAL}
DISP allows you to control how JES disposes of the data that is created before the SETPRT request. This parameter is valid only for SYSOUT data sets and is ignored for the direct user who issues SETPRT. You may abbreviate the parameters to S, N, and E, respectively. This operand is valid for the 3800 printer only.

SCHEDULE
: specifies that JES is to schedule the previous data for printing immediately.

NOSCHEDULE
: specifies that JES is to separate the data into a separate JES data set and to schedule the previous data set for printing after the job terminates.

EXTERNAL
: specifies that the schedule of the data set for printing is determined by the JCL parameter FREE=CLOSE. FREE=CLOSE is the same as specifying DISP=SCHEDULE. The absence of FREE=CLOSE in the JCL is the same as coding DISP=NOSCHEDULE on the SETPRT macro. EXTERNAL is the default.

FCB={imageid|A(address)|R(register)**}**
{({imageid|A(address)|R(register)**},{V|A})}**
The **FCB** operand specifies that the forms control buffer (FCB) is to be selected from the image library. When the **FCB** operand is specified, the **OPTCD** operand can also be specified. The possible specifications are:

imageid
: specifies the forms control image to be loaded. A forms control image is identified by a 1- to 4-character name. IBM-supplied 3211 format images are identified by imageid value of STD1 and STD2; user-designed forms control images are defined by the installation. For descriptions of the standard forms control images for the 3203 and 3211, see System-Data Administration. For more information about 3800 FCB modules, see Utilities.

A(address)
: The address subparameter specifies an in-storage address of the user-supplied forms control buffer module to be used. (For information on the format of the module, see Utilities.) This subparameter is valid for the 3800 Model 1 printer only.

R(register)
: The register subparameter specifies the register that contains an in-storage address of the user-provided forms control buffer module to be used when printing a data set. (For information on the format of the module, see Utilities.) This subparameter is valid for the 3800 Model 1 printer only.

V or **VERIFY**
: requests that the forms control image be displayed on the printer for visual verification. This operand allows forms verification and alignment using the WTOR macro instruction.

A or **ALIGN**
: allows forms alignment using the WTOR macro instruction. This subparameter will be ignored if specified for the 3800 printer.

FLASH={name}
{([name],count)}

The FLASH operand identifies the forms overlay frame to be used. Unless REXMIT=Y is coded and the forms overlay frame is still in use from the previous SETPRT macro issuance, a message tells the operator to insert this forms overlay frame into the printer. This operand also enables you to specify the number of copies on which the overlay is to be printed (flashed). If this operand is omitted, flashing ceases. This operand is valid for the 3800 printer only.

name
: is the 1- to 4-character name of the forms overlay frame.

count
: indicates the total number (0 to 255) of copies of each page of the data set on which the overlay will be printed, beginning with the first copy. The number of copies printed will not be greater than the number of copies specified by the COPIES operand. No copies will be flashed if the count of zero is specified. If a nonzero count is specified and the name of the forms overlay frame is omitted, the operator will not be requested to insert a frame. Whatever frame is inserted will be used.

INIT={N|Y}

INIT=Y will initialize the control information in the 3800 printer with a folded character arrangement table, the 10-pitch Gothic character set (12 pitch for the 3800 Model 3), and a six lines per inch FCB corresponding to the forms size in the printer. COPIES and COPYNR will be initialized to 1, FLASH and MODIFY will be initialized to none, and BURST will be initialized to N (continuous forms). For INIT=N, all control information for the 3800 printer will remain unchanged. Any parameters included on the same macro statement as the INIT operand will be processed after printer initialization has been completed. This operand is valid for the 3800 printer only.

LIBDCB=dcbaddress—A-Type Address or (2-12)

dcbaddress is the address of an authorized user library DCB that has been opened, and that you want to use instead of SYS1.IMAGELIB. If LIBDCB is not specified, SYS1.IMAGELIB is used. This operand is valid with 3800 Model 1 only.

Note: This is for the user with direct control of the 3800 Model 1.

MODIFY={name**|A(**address**)|R(**register**)}**
{({name**|A(**address**)|R(**register**)},**trc**)}**
The MODIFY operand identifies the copy modification module and an associated character arrangement table module to be used when modifying the data to be printed. This operand is valid for the 3800 Model 1 printer only.

name
A 1- to 4-character name of the copy modification module stored in SYS1.IMAGELIB. These one to four characters are the last characters of the 8-byte member name of a copy modification module in SYS1.IMAGELIB.

A(address)
The address subparameter specifies an in-storage address of the user-supplied copy modification module. This subparameter is valid for the 3800 Model 1 printer only. For information on the format of the module, see Utilities.

R(register)
The register subparameter specifies a register that contains an in-storage address of the user-provided copy modification module. This subparameter is valid for the 3800 Model 1 printer only. For information on the format of the module, see Utilities.

trc
specifies the table reference character used to select one of the character arrangement table modules to be used for the copy modification text. The values of 0, 1, 2, and 3 correspond to the order in which the module names have been specified in the **CHARS** operand. If trc is not included, the first character arrangement table module (0) is assumed.

MSGAREA=address—A-Type Address or (2-12)
address is the address of the message feedback area. This area is used to transfer message text between the SETPRT macro and the caller. You must allow at least 80 bytes for the message text plus 10 bytes for prefix information or a total length of at least 95 bytes. The message is truncated if it does not fit into the area. This operand is valid with 3800 only. The following shows the layout of the message area:

bytes 0-1:	total length
bytes 2-5:	reserved
bytes 6-7:	text length
bytes 8-9:	reserved
bytes 10-variable:	message text

OPTCD={B|U}
{({B|U},{F|U})}
The **OPTCD** operand specifies whether printer data checks are blocked or unblocked and if the printer is to operate in fold or normal mode. The possible specifications are:

B
specifies that printer data checks are blocked; this option updates the DCBOPTCD field of the data control block.

U
specifies that printer data checks are unblocked; this option updates the DCBOPTCD field of the data control block.

F or **FOLD**
specifies that printing is in fold mode. This subparameter is ignored if specified for the 1403 or 3800 printer. For 1403 fold mode, use fold option under the UCS operand.

U or **UNFOLD**
specifies that printing is in normal mode; this operand causes fold mode to revert to normal mode. This subparameter will be ignored if specified for the 1403 or 3800 printer.

PRTMSG={N|Y}
PRTMSG allows printer error messages to be printed for the programmer on the 3800. This operand is valid with 3800 only.

N
specifies not to print error messages on the 3800.

Y
specifies to print error messages on the 3800. Y is the default.

REXMIT={N|Y}
The specification of REXMIT=Y allows modification of the starting copy number (COPYNR), the number of copies of the pages in a data set to be printed (COPIES), the forms overlay frame to be used (FLASH), and the number of copies to be printed (FLASH) without changing the other control information already set up in the printer. The SETPRT SVC will ignore all other parameters in the parameter list.

UCS={csc}
{(csc,{F|F,V|V})}
The **UCS** operand specifies that the UCS buffer is to be loaded from the image library. When the **UCS** operand is specified, the **FCB** and **OPTCD** operands can also be specified. This operand will be ignored if specified for the 3800 printer. The possible specifications are:

csc (character set code)
The csc operand specifies the character set selected. A character set is identified by a 1- to 4-character code. Codes for standard IBM character sets are as follows:

1403 or **3203 Printer: AN, HN, PCAN, PCHN, PN, QN, QNC, RN, SN, TN, XN,** and **YN**

3211 Printer: A11, H11, G11, P11, and **T11**

For descriptions of the standard IBM character sets, see System Generation; codes for user-designed character sets are defined by the installation. For information on adding user-defined entries to an image table, see System-Data Administration.

F or **FOLD**
specifies that the character set image selected is to be in fold mode. The fold mode translates the EBCDIC code for lowercase characters to the EBCDIC code for the corresponding uppercase characters.

V or **VERIFY**
requests that the character set image be displayed on the printer for visual verification.

RETURN CODES

After the SETPRT macro instruction is executed, a return code is placed in register 15, and control is returned to the instruction following the SETPRT macro instruction. The illustration below shows how the four bytes of register 15 are used for a specific printer.

Byte	0	1	2	3
	Unused	3800 Code Other than FCB	FCB Code	UCS Code
Bit	0 7	8 15	16 23	24 31

The return codes in the figures that follow are in hexadecimal.

- Return codes 0 through 14 apply to a specific printer.
- Return codes 18 through 24 apply to all printers.
- Return codes 28 through 50 apply to the 3800 printer only.

Figure 1 shows the hexadecimal return codes 00 through 14 for specific printers. An 'XX' in the columns labeled '3800 Code Other than FCB' or 'FCB Code,' means that a nonzero code may be in that byte.

3800 Code Other than FCB (Byte 1)	FCB Code (Byte 2)	UCS Code (Byte 3)	Meaning
00	00	00	Successful completion.
00	XX	04	The operator canceled the UCS request for the following reason: • The UCS image could not be found in SYS1.IMAGELIB.
00	04	XX	For non-3800 printers, the operator canceled the FCB load operation for one of the following reasons: • The form could not be aligned to match the buffer. • The FCB module could not be found in SYS1.IMAGELIB or the user's DCB exit list. For a 3800, the specified FCB module could not be found in SYS1.IMAGELIB, a user library, or the DCB exit list, and SETPRT processing was terminated.

Figure 1 (Part 1 of 3). SETPRT Return Codes for Specific Printers

3800 Code Other than FCB (Byte 1)	FCB Code (Byte 2)	UCS Code (Byte 3)	Meaning
04	00	00	The 3800 SETPRT processing was suspended for one of the following reasons: • A character arrangement table module could not be found in SYS1.IMAGELIB or a user library. • A copy modification module could not be found in SYS1.IMAGELIB or a user library. • A graphic character modification module (required by a character arrangement table module) could not be found in SYS1.IMAGELIB or a user library. • A library character set module could not be found in SYS1.IMAGELIB or a user library. Register 0 contains a reason code identifying which of the above conditions occurred. For an explanation, see Figure 3 on page 157.
00	XX	08	A permanent I/O error was detected when the BLDL macro instruction was issued to locate a UCS image in SYS1.IMAGELIB.
00	08	XX	A permanent I/O error was detected when the BLDL macro instruction was issued to locate an FCB module in SYS1.IMAGELIB or a user library.
08	00	00	A permanent I/O error was detected when the BLDL macro instruction was issued to locate one of the following modules in SYS1.IMAGELIB or a user library. • A character arrangement table module • A copy modification module • A graphic character modification module • A library character set module Register 0 contains a reason code identifying which of the above conditions occurred. For an explanation, see Figure 3 on page 157.
00	XX	0C	A permanent I/O error was detected while loading the printer's UCS buffer.
00	0C	XX	A permanent I/O error was detected during forms positioning or while loading the printer's FCB buffer. Register 0 contains a reason code identifying which of the above conditions occurred. For an explanation, see Figure 3 on page 157.

Figure 1 (Part 2 of 3). SETPRT Return Codes for Specific Printers

3800 Code Other than FCB (Byte 1)	FCB Code (Byte 2)	UCS Code (Byte 3)	Meaning
0C	00	00	A permanent I/O error was detected while loading one of the following: • Character arrangement table • Copy modification record • Starting copy number • Graphic character modification record • Forms overlay sequence control record (copy counts and flash counts) • Writable character generation module (WCGM) • Library character set (3800 only) Register 0 contains a reason code identifying which of the above conditions occurred. For an explanation, see Figure 3 on page 157.
00	XX	10	A permanent I/O error was detected when an attempt was made to display the character set image on the printer for visual verification.
00	10	XX	A permanent I/O error was detected when an attempt was made to display the forms control image on the printer for visual verification.
00	XX	14	The operator canceled the UCS request because an improper character set image was displayed for visual verification.
00	14	XX	The operator canceled the FCB request because an improper forms control image was displayed for visual verification.

Figure 1 (Part 3 of 3). SETPRT Return Codes for Specific Printers

The illustration below shows how the four bytes of register 15 are used for all printers.

Byte	0	1	2	3
	Unused	Unused	Unused	General Code
Bit	0 7	8 15	16 23	24 31

Figure 2 shows the hexadecimal return codes 18 through 24 for all printers.

Return Code (Byte 3)	Meaning
18	No operation was performed for one of the following reasons: • The data control block was not open • The data control block was not valid for a sequential data set. • The SETPRT parameter list was not valid • The output device was not a UCS or 3800 printer
1C	No operation was performed because an uncorrectable error occurred in a previously initiated output operation. The error analysis (SYNAD) routine is entered when the next PUT or CHECK macro instruction is issued. No operation was performed because an uncorrectable error occurred when the block data check or the reset block data check command was issued by SETPRT. For a 3800, message IEC173I will indicate which of the above errors has occurred. Register 0 contains a reason code identifying whether or not data was lost. For an explanation, see Figure 4 on page 157.
20	Not enough space has been provided for the SYS1.IMAGELIB or a user library control blocks.
24	SYS1.IMAGELIB (or, for the 3800, a user library) cannot be opened to load the specified module.
28	The operator canceled the forms overlay request.
2C	The operator canceled the paper threading request.
30	There are more writable character generation modules (WCGMs) requested than there are writable buffers installed on the printer.
34	There was an invalid table reference character for copy modification.
38	An error occurred when attempting to execute the initialize printer command.
3C	Bursting was requested but the Burster-Trimmer-Stacker feature is not installed on the printer.
40	A permanent I/O error occurred while executing a sense, final select character arrangement table command, or display status code.
44	The translate table character arrangement table entry references a character set that is not in the image library.

Figure 2 (Part 1 of 2). SETPRT Return Codes for All Printers

Return Code (Byte 3)	Meaning
48	Data was lost because of one of the following (3800 only): • 3800 system restart after a paper jam • Cancel key • Lost resources after paper jam
4C	A load check was detected while loading one of the following (3800 only): • Forms control buffer (FCB) • Character arrangement table (CAT) • Graphic arrangement table (GCM) • Copy modification record • Writable character generation module (WCGM) • Library character set (LCS) Register 0 contains a reason code identifying which of the above conditions occurred. For an explanation, see Figure 3 on page 157.
50	When a SETPRT was issued to a SYSOUT data set, there was a failure in one of the following (3800 only): • The subsystem interface (SSI) for OPEN or CLOSE • Data set segmentation • Queue manager issuing I/O to read the JFCB and/or the JFCBE • ENQ failure • More than one DCB is open for the SYSOUT data set

Figure 2 (Part 2 of 2). SETPRT Return Codes for All Printers

IBM 3800 MODELS 1 AND 3

These reason codes, returned in register 0, for the 3800 Models 1 and 3 printers are in addition to completion codes 04, 08, 0C, and 4C returned in register 15.

The following illustration shows the contents of register 0, which includes the GCM ID, the CAT ID, and the reason code.

Byte	0	1	2	3
	Unused	GCM ID	CAT ID	Reason Code
Bit	0 7	8 15	16 23	24 31

Figure 3 shows the hexadecimal reason codes for the IBM 3800 Model 1 and the IBM 3800 Model 3 in compatibility mode.

GCM ID (Byte 1)	CAT ID (Byte 2)	Reason Code (Byte 3)	Meaning
00	01-04	04	Character arrangement table module/record
00	00	08	Copy modification module/record
00	00	0C	Starting copy number
01-04	01-04	10	Graphic character modification module/record
00	00	14	Forms overlay sequence control record
00	00	18	Library character set
00	00	1C	Writable character generation module (WCGM)
00	00	20	Forms control buffer module

Figure 3. Reason Codes for IBM 3800 Models 1 and 3 Printers (GCM ID and CAT ID)

Figure 4 shows the reason codes in addition to return code 1C returned in register 15. The reason code is placed in byte 3 of register 0. Bytes 0 through 2 are reserved.

Reason Code (Byte 3)	Meaning
00	Indicates no data lost.
04	Indicates data has been lost.

Figure 4. Reason Codes for All Printers (Bytes 0 through 2 Reserved)

Figure 5 on page 158 shows the reason codes in addition to return code 48 returned in register 15. The reason code is placed in byte 3 of register 0. Bytes 0 through 2 are reserved.

Reason Code (Byte 3)	Meaning
04	A paper jam caused a restart. A possible lost data condition was detected.
08	The cancel key was pressed.
0C	Resources were lost after a paper jam.

Figure 5. Reason Codes for IBM 3800 Models 1 and 3 Printers (Bytes 0 through 2 Reserved)

Figure 6 shows the reason codes in addition to return code 50 returned in register 15. The reason code is placed in byte 3 of register 0. Bytes 0 through 2 are reserved.

Reason Code (Byte 3)	Meaning
04	An invalid SETPRT request for a SYSOUT data segment was specified. An in-storage address was used for a copymod, character arrangement table, FCB, or user library DCB. Only 3800 load module IDs in SYS1.IMAGELIB are allowed for SYSOUT setup for 3800. If Direct Attach (a directly allocated 3800 Model 3 or Model 8 printer) is active, data management treats the device specified by the **UNIT** parameter as a SYSOUT data set.
08	During SETPRT processing for a SYSOUT data segment, an error was detected while attempting to read a JFCB or JFCBE control block from SWA
0C	During SETPRT processing for a SYSOUT data segment, an error was detected while invoking the CLOSE subsystem interface (SSI) for the previous data segment
10	During SETPRT processing for a SYSOUT data segment, an error was detected while invoking the OPEN subsystem interface (SSI) for the new data segment being created
14	During SETPRT processing for a SYSOUT data segment, an error was detected while the scheduler spool file allocation routine was segmenting the data set
18	An ENQ macro failed. The ENQ was issued by SETPRT processing.
1C	More than one DCB is open for the SYSOUT data set.

Figure 6. Reason Codes for IBM 3800 Models 1 and 3 Printers (Bytes 0 through 2 Reserved)

SETPRT—LIST FORM

The list form of the SETPRT macro instruction is used to construct a data management parameter list.

The description of the standard form of the SETPRT macro instruction provides the explanation of the function of each operand. The format description below indicates the optional and required operands for the list form only. The dcbaddr parameter must appear in the list or execute form of the SETPRT macro.

The list form of the SETPRT macro instruction is written as follows:

[symbol]	SETPRT	[dcbaddr]
		[,BURST={N|Y}]
		[,CHARS={[name}
		{(name,...)}]
		[,COPIES=number]
		[,COPYNR=number]
		[,DISP=[SCHEDULE|NOSCHEDULE|EXTERNAL]
		[,FCB={imageid}
		{(imageid,{V|A})}]
		[,FLASH={name}
		{([name],count)}]
		[,INIT={N|Y}]
		[,LIBDCB=dcbaddress]
		[,MODIFY={name}
		{(name,trc)}]
		[,MSGAREA=address]
		[,OPTCD={B|U}
		{({B|U},{F|U})}]
		[,PRTMSG=[N|Y]
		[,REXMIT={N|Y}]
		[,UCS={csc}
		{(csc,{F|F,V|V})}]
		,MF=L

dcbaddr—A-Type Address

BURST={N|Y}
is coded as shown in the standard form of the macro instruction.

CHARS={name}
{(name,...)}
is coded as shown in the standard form of the macro instruction, except for the A(address) and R(register) parameters, which cannot be specified.

COPIES=number
is coded as shown in the standard form of the macro instruction.

COPYNR=number
is coded as shown in the standard form of the macro instruction.

DISP={SCHEDULE|NOSCHEDULE|EXTERNAL}
is coded as shown in the standard form of the macro instruction.

FCB={imageid}
{(imageid,{V|A})}
is coded as shown in the standard form of the macro instruction, except for the A(address) and R(register) parameters, which cannot be specified.

FLASH={name}
{([name],count)}
is coded as shown in the standard form of the macro instruction.

INIT={N|Y}
is coded as shown in the standard form of the macro instruction.

LIBDCB=dcbaddress—A-Type Address or (2-12)
is coded as shown in the standard form of the macro instruction.

MODIFY={name}
{(name,trc)}
is coded as shown in the standard form of the macro instruction, except for the A(address) and R(register) parameters, which cannot be specified.

MSGAREA=address—A-Type Address or (2-12)
is coded as shown in the standard form of the macro instruction.

OPTCD={B|U}
{({B|U},{F|U})}
is coded as shown in the standard form of the macro instruction.

PRTMSG={N|Y}
is coded as shown in the standard form of the macro instruction.

REXMIT={N|Y}
is coded as shown in the standard form of the macro instruction.

UCS={csc}
{(csc,{F|F,V|V})}
is coded as shown in the standard form of the macro instruction.

MF=L
specifies that the list form of the macro instruction is used to create a parameter list that can be referenced by an execute form of the SETPRT macro instruction.

SETPRT—EXECUTE FORM

A remote data management parameter list is referred to, and can be modified by, the execute form of the SETPRT macro instruction.

The description of the standard form of the SETPRT macro instruction provides the explanation of the function of each operand. The format description below indicates the optional and required operands for the execute form only. The dcbaddr parameter must be specified in the list or execute form of the SETPRT macro.

The execute form of the SETPRT macro instruction is written as follows:

```
[symbol]  SETPRT  [dcbaddr]
                  [,BURST={N|Y|*}]
                  [,CHARS={name|A(address)|R(register)}
                          {({name|A(address)|R(register)},...)}
                          {*}]
                  [,COPIES={number|*}]
                  [,COPYNR={number|*}]
                  [,DISP=[SCHEDULE|NOSCHEDULE|EXTERNAL]
                  [,FCB={imageid|A(address)|R(register)}
                        {({imageid|A(address)|R(register),
                        {V|A})}[*}]
                  [,FLASH={name}
                          {([name],count)}
                          {*}]
                  [,INIT={N|Y}]
                  [,LIBDCB=dcbaddress]
                  [,MODIFY={name|A(address)|R(register)}
                           {({name|A(address)|R(register)},trc}
                           {*}]
                  [,MSGAREA=address]
                  [,OPTCD={B|U}
                          {({B|U},{F|U})}]
                  [,PRTMSG=[N|Y]
                  [,REXMIT={N|Y|*}]
                  [,UCS={csc}
                        {(csc,{F|F,V|V})}]
                  ,MF=(E,{data management list address|(1)})
```

dcbaddr—RX-Type Address or (2-12)

BURST={N|Y|*}
is coded as shown in the standard form of the macro instruction, except for the * subparameter, which can be used only when INIT=Y is specified in the execute form of the SETPRT macro instruction. When BURST=* is coded, the BURST field in the parameter list remains as it was previously set. This operand is valid for the 3800 printer only.

CHARS={name|A(address)|R(register)}
{({name|A(address)|R(register)},...)}
{*}
is coded as shown in the standard form of the macro instruction, except for the * subparameter, which can be used only when INIT=Y is specified in the execute form of the SETPRT macro instruction. When CHARS=* is coded, the CHARS field in the parameter list remains as it was previously set.

COPIES={number|*}
is coded as shown in the standard form of the macro instruction, except for the * subparameter, which can be used only when INIT=Y is specified in the execute form of the SETPRT macro instruction. When COPIES=* is coded, the COPIES field in the parameter list remains as it was previously set.

COPYNR={number|*}
is coded as shown in the standard form of the macro instruction, except for the * subparameter, which can be used only when INIT=Y is specified in the execute form of the SETPRT macro instruction. When COPYNR=* is coded, the COPYNR field in the parameter list remains as it was previously set.

DISP={SCHEDULE|NOSCHEDULE|EXTERNAL}
is coded as shown in the standard form of the macro instruction.

FCB={imageid|A(address)|R(register)}
{({imageid|A(address)|R(register)},{V|A})}
{*}
is coded as shown in the standard form of the macro instruction, except for the * subparameter, which can be used only when INIT=Y is specified in the execute form of the SETPRT macro instruction. When FCB=* is coded, the FCB field in the parameter list remains as it was previously set.

FLASH={name}
{([name],count)}
{*}
is coded as shown in the standard form of the macro instruction, except for the * subparameter, which can be used only when INIT=Y is specified in the execute form of the SETPRT macro instruction. When FLASH=* is coded, the FLASH field in the parameter list remains as it was previously set.

INIT={N|Y}
is coded as shown in the standard form of the macro instruction. When INIT=Y is specified on the execute form of the SETPRT macro instruction, all 3800 fields in the parameter list (BURST, CHARS, COPIES, COPYNR, FCB, FLASH, MODIFY, and REXMIT) will be reset to binary zeros unless a specified field is preserved by coding keyword parameter=* or changed by specifying a valid subparameter for the keyword parameter as described in the standard form of the macro instruction.

LIBDCB=dcbaddress—A-Type Address or (2-12)
is coded as shown in the standard form of the macro instruction.

MODIFY={name|A(address)|R(register)}
{({name|A(address)|R(register)},trc)}
{*}
is coded as shown in the standard form of the macro instruction, except for the * subparameter, which can be used only when INIT=Y is specified in the execute form of the SETPRT macro instruction. When MODIFY=* is coded, the MODIFY field in the parameter list remains as it was previously set.

MSGAREA=address—A-Type Address or (2-12)
is coded as shown in the standard form of the macro instruction.

OPTCD={B|U}
{({B|U},{F|U})}
is coded as shown in the standard form of the macro instruction.

PRTMSG={N|Y}
is coded as shown in the standard form of the macro instruction.
is coded as shown in the standard form of the macro instruction.

REXMIT={N|Y|*}
is coded as shown in the standard form of the macro instruction, except for the * subparameter, which can be used only when INIT=Y is specified in the execute form of the SETPRT macro instruction. When REXMIT=* is coded, the REXMIT field in the parameter list remains as it was previously set.

UCS={csc}
{(csc,{F|F,V|V})}
is coded as shown in the standard form of the macro instruction.

MF=(E,{data management list address|(1)})
specifies that the execute form of the SETPRT macro instruction is used, and an existing data management parameter list is used.

E—Coded as shown

data management list address—RX-Type Address, (2-12), or (1)

STOW—UPDATE PARTITIONED DATA SET DIRECTORY (BPAM)

The STOW macro instruction causes the system to update a partitioned data set directory by adding, changing, replacing, or deleting an entry in the directory. Only one entry can be updated at a time using the STOW macro instruction. If the entry to be added or replaced is a member name, the system writes an end-of-data indication following the member. All input/output operations using the same data control block must have previously been tested for completion.

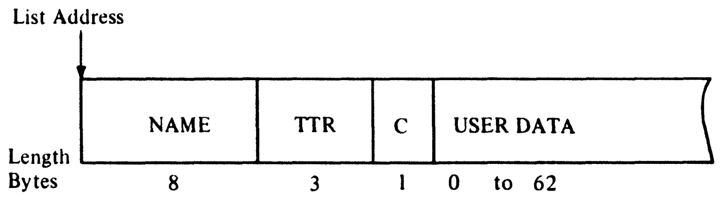
The STOW macro is written:

[symbol]	**STOW**	dcb address ,list address [,directory action] [,directory action[A|C|D|R]]

dcb address—RX-Type Address, (2-12), or (1)
The dcb address operand specifies the address of the data control block for the opened partitioned data set. The STOW macro instruction can be used only when the data set is opened for OUTPUT, UPDAT or OUTIN (BSAM).

list address—RX-Type Address, (2-12), or (0)
The list address operand specifies the address of the area containing the information required by the system to maintain the partitioned data set directory. The size and format of the area depend on the directory action requested as follows:

Adding or Replacing a Directory Entry: The list address operand must specify an area at least 12 bytes long and beginning on a halfword boundary. The following illustration shows the format of the area:



NAME: Specifies the member name or alias being added or replaced. The name must begin in the first byte of the field and be padded on the right with blanks, if necessary, to complete the 8-byte field.

TT: Specifies the relative track number where the beginning of the data set is located.

R: Specifies the relative block (record) number on the track identified by **TT**.

Note: The TTR field shown above must be supplied by the problem program if an alias (alias bit is 1) is being added or replaced. The system supplies the TTR field when a member name is being added or replaced.

C: Specifies the type of entry (member or alias) for the name, the number of note list fields (TTRNs), and the length in halfwords, of the user data field. The following describes the meaning of the 8 bits:

Bit	**Meaning**
0=0	Indicates a member name.
0=1	Indicates an alias.
1 and 2	Indicate the number of TTRN fields (maximum of 3) in the user data field.
3-7	Indicate the total number of halfwords in the user data field.

Deleting a Directory Entry: The list address operand must specify an 8-byte area that contains the member name or alias to be deleted. The name must begin in the first byte of the area and be padded on the right with blanks, if necessary, to complete the 8 bytes.

Changing the Name of a Member: The list address operand must specify the address of a 16-byte area; the first 8 bytes contain the old member name or alias, and the second 8 bytes contain the new member name or alias. Both names must begin in the first byte of their 8-byte area and be padded on the right with blanks, if necessary, to complete the 8-byte field.

<u>directory action</u>—**[def.A|C|D|R]**
If the directory action operand is not coded, A (add an entry) is the default. The operand is coded as shown to specify the type of directory action:

A
specifies that an entry is to be added to the directory.

C
specifies that the name of an existing member or alias is to be changed.

D
specifies that an existing directory entry is to be deleted.

R
specifies that an existing directory entry is to be replaced by a new directory entry. If R is coded but the old entry is not found, the new entry is added to the directory and a completion code of X'08' is returned in register 15.

COMPLETION CODES

When the system returns control to the problem program, register 15 contains a return code and register 0 contains a reason code in the low-order byte; the three high-order bytes of both registers are set to 0.

The following is a list of return codes contained in register 15:

Codes	Directory Action			
Return (15)	A	R	D	C
00	The update of the directory was completed successfully.	The update of the directory was completed successfully.	The update of the directory was completed successfully.	The update of the directory was completed successfully.
04	The directory already contains the specified name.	—	—	The directory already contains the specified new name.
08	—	The specified name could not be found.	The specified name could not be found.	The specified old name could not be found.
0C	No space left in the directory. The entry could not be added, replaced, or changed.	No space left in the directory. The entry could not be added, replaced, or changed.	—	No space left in the directory. The entry could not be added, replaced, or changed.
10	A permanent input or output error was detected. Control is not given to the error analysis (SYNAD) routine.	A permanent input or output error was detected. Control is not given to the error analysis (SYNAD) routine.	A permanent input or output error was detected. Control is not given to the error analysis (SYNAD) routine.	A permanent input or output error was detected. Control is not given to the error analysis (SYNAD) routine.
14	The specified data control block is not open or is opened for input.	The specified data control block is not open or is opened for input.	The specified data control block is not open or is opened for input.	The specified data control block is not open or is opened for input.
18	Insufficient virtual storage was available to perform the STOW function.	Insufficient virtual storage was available to perform the STOW function.	Insufficient virtual storage was available to perform the STOW function.	Insufficient virtual storage was available to perform the STOW function.

The following is a list of reason codes contained in register 0.

Reason Code (0)	Meaning
00 (X'00')	Reason code is not applicable. (Returned with all return codes except 10.)
01 (X'01')	All functions; the permanent I/O error occurred while reading or writing directory blocks.
02 (X'02')	Add and replace functions; the permanent I/O error occurred while EOF mark after the member.
3383 (X'D37')	Error occurred when trying to write an EOF; all primary space used.

SYNADAF—PERFORM SYNAD ANALYSIS FUNCTION (BDAM, BISAM, BPAM, BSAM, EXCP, QISAM, AND QSAM)

The SYNADAF macro instruction is used in an error analysis routine to analyze permanent input/output errors. The routine can be a SYNAD exit routine specified in a data control block for BDAM, BISAM, BPAM, BSAM, QISAM, QSAM, or a routine that is entered directly from a program that uses the EXCP macro instruction. (The EXCP macro instruction is described in _Data Administration Guide_.)

The SYNADAF macro instruction uses register 1 to return the address of a buffer containing a message. The message describes the error, and can be printed by a subsequent PUT or WRITE macro instruction. The message consists of EBCDIC information and is in the form of a variable-length record. The format of the message is shown following the descriptions of the SYNADAF operands.

The system does not use the save area whose address is in register 13. Instead, it provides a save area for its own use, and then makes this area available to the error analysis routine. The system returns the address of the new save area in register 13 and in the appropriate location (word 3) of the previous save area; it also stores the address of the previous save area in the appropriate location (word 2) of the new save area.

The SYNADAF macro instruction passes parameters to the system in registers 0 and 1. When used in a SYNAD exit routine, the SYNADAF macro should be coded at the beginning of the routine. (See _Data Facility Product: Customization_.) For BISAM and QISAM, the SYNAD exit routine has to set up these parameters as explained under PARM1 and PARM2. To save these parameters for use by the SYNAD exit routine, the system stores them in a parameter save area that follows the message buffer as shown in the message buffer format. The system does not alter the return address in register 14 or the entry point address in register 15.

When a SYNADAF macro instruction is used, a SYNADRLS macro instruction must be used to release the message buffer and save areas, and to restore the original contents of register 13.

The SYNADAF macro is written:

```
[symbol]   SYNADAF    ACSMETH={BDAM
                                [,PARM1=parm register]
                                [,PARM2=parm register]}
                               {BPAM
                                [,PARM1=parm register]
                                [,PARM2=parm register]}
                               {BSAM
                                [,PARM1=parm register]
                                [,PARM2=parm register]}
                               {QSAM
                                [,PARM1=parm register]
                                [,PARM2=parm register]}
                               {BISAM
                                [,PARM1=dcbaddr]
                                [,PARM2=dcb address]}
                               {EXCP
                                [,PARM1=iob address]}
                               {QISAM
                                [,PARM1=dcbaddr]
                                [,PARM2=parm register]}
```

ACSMETH=BDAM, BPAM, BSAM, QSAM, BISAM, EXCP, or QISAM
specifies the access method used to perform the input/output operation for which error analysis is performed.

PARM1=parm register, iobaddr, or dcbaddr—(2-12) or (1)
specifies the address of information that is dependent on the access method being used. For BDAM, BPAM, BSAM, or QSAM, the operand specifies a register that contains the information that was in register 1 on entry to the SYNAD routine. For BISAM or QISAM, it specifies the address of the data control block; for EXCP, it specifies the address of the input/output block. If the operand is omitted, **PARM1=(1)** is assumed.

PARM2=parm register—(2-12), (0), or RX-Type
(only if **ACSMETH=QISAM**)
specifies the address of additional information that is dependent on the access method being used. For BDAM, BPAM, BSAM, QISAM, and QSAM, the operand specifies a register that contains the information that was in register 0 on entry to the SYNAD exit routine. For BISAM, the operand specifies a register that contains the information that was in register 1 on entry to the SYNAD exit routine (the address of the DECB). For EXCP, the operand is meaningless and should be omitted. If the operand is omitted, except in the case of EXCP, **PARM2=(0)** is assumed.

Note: To correctly load the registers for SYNADAF for BISAM, code these two instructions before issuing the SYNADAF macro:

```
LR   0,1          GET DECB ADDRESS

L    1,8(1)       GET DCB ADDRESS
```

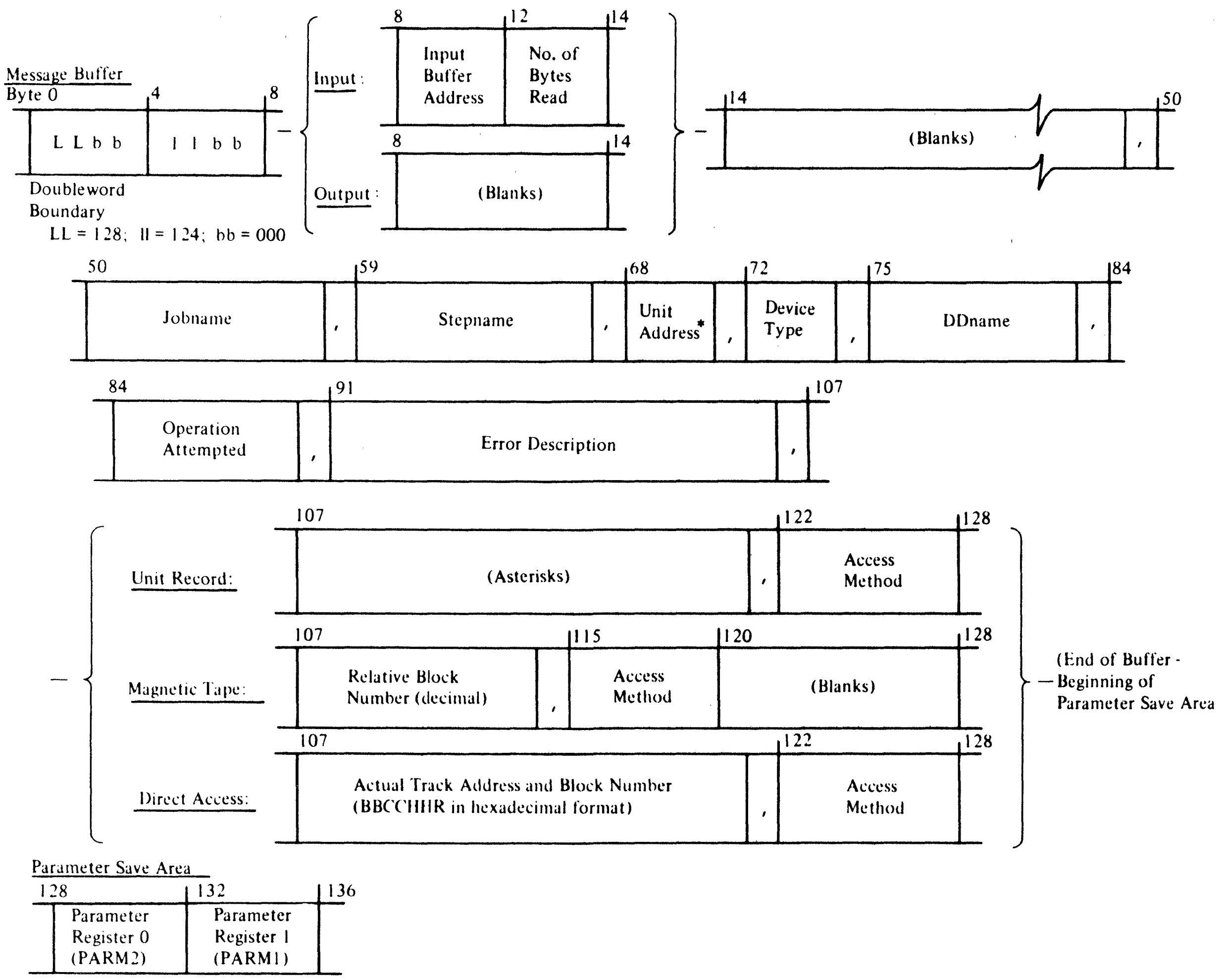
COMPLETION CODES

When the system returns control to the problem program, the low-order byte of register 0 contains one of the following reason codes; the three high-order bytes of register 0 are set to 0.

Reason Code (0)	Meaning
00 (X'00')	Successful completion. Bytes 8 through 13 of the message buffer contain blanks.
04 (X'04')	Successful completion. Bytes 8 through 13 of the message buffer contain binary data.
08 (X'08')	Unsuccessful completion. The message can be printed, but some information is missing in bytes 50 through 127 and is represented by asterisks. If byte 8 is a blank (X'40'), bytes 9 through 13 are either blanks or are not initialized. If byte 8 is not a blank, then data was read, and bytes 8 through 13 of the message buffer contain binary data.

MESSAGE BUFFER FORMAT

The following illustration shows the format of the message buffer; the address of the buffer is returned in register 1.



Notes:

1. The device type field (bytes 72 through 73) contains UR for a unit record device, TA for a magnetic tape device, or DA for a direct access device.

2. If a message field (bytes 91 through 105) is not applicable to the type of error that occurred, it contains N/A or NOT APPLICABLE.

3. If no data was transmitted, or if the access method is QISAM, bytes 8 through 13 contain blanks or binary zeros.

4. If the access method is BISAM, bytes 68 through 70, 84 through 89, and 107 through 120 contain asterisks.

5. If the access method is BDAM, and if the error was an invalid request, bytes 107 through 120 contain EBCDIC zeros.

6. The unit address field (bytes 68 through 70) contains the letters 'JES' if the data set is SYSIN or SYSOUT.

SYNADRLS—RELEASE SYNADAF BUFFER AND SAVE AREAS (BDAM, BISAM, BPAM, BSAM, EXCP, QISAM, AND QSAM)

The SYNADRLS macro instruction releases the message buffer, parameter save area, and register save area provided by a SYNADAF macro instruction. It must be used to perform this function whenever a SYNADAF macro instruction is used.

When the SYNADRLS macro instruction is issued, register 13 must contain the address of the register save area provided by the SYNADAF macro instruction. The control program loads register 13 with the address of the previous save area, and sets word 3 of that save area to 0. Thus, when control is returned, the save area pointers are the same as before the SYNADAF macro instruction was issued.

The SYNADRLS macro is written:

[symbol]	SYNADRLS	b

When the system returns control to the problem program, the low-order byte of register 0 contains one of the following reason codes; the three high-order bytes of register 0 are set to 0.

Reason Code (0)	Meaning
00 (X'00')	Successful completion.
08 (X'08')	Unsuccessful completion. The buffer and save areas were not released; the contents of register 13 remain unchanged. Register 13 does not point to the save area provided by the SYNADAF macro instruction, or this save area is not properly chained to the previous save area.

SYNCDEV—SYNCHRONIZE DEVICE

The SYNCDEV macro instruction allows you to control data synchronization for the IBM 3480 Magnetic Tape Subsystem that supports buffered write mode. Data records in the tape control unit buffer may not yet be on tape when your program is ready to send more. There is no way to determine how much data is left in the buffer, and it is time dependent to tape motion. This data is not synchronized to your program; that is, you could overlay unwritten data in the buffer, or lose data when it is transferred from the channel if the buffer does not have enough space to hold it. You can use the SYNCDEV macro to either:

- Request information regarding synchronization
- Demand synchronization if the specified number of data blocks are buffered

  If more blocks are buffered than were specified, the system stays in control until all the blocks are written to the tape or it detects an I/O error.

  If the same amount or fewer blocks are buffered, buffering is not affected.

Note: Demands for synchronization are ignored if the drive is in read mode.

The SYNCDEV macro is written:

[<u>symbol</u>]	**SYNCDEV**	**DCB=**<u>addr</u> **[,{ABUFBLK=**<u>addr</u>**|BUFBLK=** **{**<u>maximum buffer depth</u>**|**<u>0</u>**}}]** **[,INQ={YES|**<u>NO</u>**}]**

The following describes the operands that can be specified for SYNCDEV.

DCB=<u>addr</u>—A-Type address or (2-12)
specifies the address of the data control block.

ABUFBLK=<u>addr</u>**|BUFBLK=**<u>maximum buffer depth</u>**|**<u>0</u>
specifies the maximum number of data blocks that can be buffered.

ABUFBLK=<u>addr</u>—A-Type address or (2-12)
specifies the address of a halfword on a halfword boundary that contains a value that specifies the maximum number of data blocks that can be buffered.

BUFBLK={<u>maximum buffer depth</u>**|**<u>0</u>**}**
specifies the maximum number of data blocks that can be buffered. This number can be an absolute value from 0 to 65535. The BUFBLK value can be in the two low-order bytes of a register (2-12).

<u>0</u>
If neither ABUFBLK nor BUFBLK is specified, the number of data blocks that can be buffered defaults to 0, and no data blocks are buffered.

INQ={YES|<u>NO</u>**}**
specifies whether this is a request for information about the degree of synchronization or a request for synchronization.

YES
specifies an inquiry as to how many data blocks are in the buffer.

NO

specifies a request for synchronization based on the number of data blocks that can be buffered as specified in ABUFBLK or BUFBLK.

Register 0 contains the number of buffered physical blocks if the previous operation completed successfully.

Note: Do not use this option in 31-bit residence mode; it requires a 24-bit addressing mode parameter list.

SYNCDEV—LIST FORM

The list form of the SYNCDEV macro is written:

[symbol]	SYNCDEV	[DCB=addr] [,{BUFBLK=maximum buffer depth|0}] [,INQ={YES|NO}] ,MF=L

The following describes the operands that can be specified for the list form of SYNCDEV.

DCB=addr—A-Type address
specifies the address of the data control block.

BUFBLK={maximum buffer depth|0**}**
specifies the maximum number of data blocks that can be buffered. This number can be an absolute value from 0 to 65535. If BUFBLK is not specified, the number of data blocks that can be buffered defaults to 0, and no data blocks are buffered.

0
If neither ABUFBLK nor BUFBLK is specified, the number of data blocks that can be buffered defaults to 0, and no data blocks are buffered.

INQ={YES|NO}
specifies whether this is a request for information about the degree of synchronization or a request for synchronization.

YES
specifies an inquiry as to how many data blocks are in the buffer.

NO
specifies a request for synchronization based on the number of data blocks that can be buffered as specified in BUFBLK.

MF=L
generates a 24-bit addressing-mode parameter list that contains no executable instructions. The list can be used as input and can be modified by the execute form of the SYNCDEV macro.

Note: Do not use this option in 31-bit residence mode; it requires a 24-bit addressing-mode parameter list.

SYNCDEV—EXECUTE FORM

The execute form of the SYNCDEV macro is written:

[symbol]	**SYNCDEV**	**[DCB=**addr**]** **[,{ABUFBLK=**addr**|BUFBLK=** **{**maximum buffer depth**|**0**}}]** **[,INQ={YES|**NO**}]** **,MF=(E,addr)**

The following describes the operands that can be specified for the execute form of SYNCDEV.

DCB=addr—A-Type address or (2-12)
specifies the address of the data control block.

ABUFBLK=addr**\|BUFBLK={**maximum buffer depth**\|**0
specifies the maximum number of data blocks that can be buffered.

ABUFBLK=addr—A-Type address or (2-12)
specifies the address of a halfword on a halfword boundary that contains a value that specifies the maximum number of data blocks that can be buffered.

BUFBLK=maximum buffer depth**\|**0
specifies the maximum number of data blocks that can be buffered. This number can be an absolute value from 0 to 65535. The BUFBLK value can be in the two low-order bytes of a register (2-12).

0
If neither ABUFBLK nor BUFBLK is specified, the number of data blocks that can be buffered defaults to 0, and no data blocks are buffered.

INQ={YES\|NO}
specifies whether this is a request for information about the degree of synchronization or a request for synchronization.

YES
specifies an inquiry as to how many data blocks are in the buffer.

NO
specifies a request for synchronization based on the number of data blocks that can be buffered as specified in ABUFBLK or BUFBLK.

Register 0 contains the number of buffered physical blocks if the previous operation completed successfully.

Note: Do not use this option in 31-bit residence mode; it requires a 24-bit addressing mode parameter list.

MF=(E,addr**)**
specifies the execute form of SYNCDEV.

addr**—A-Type address, RX-Type address, or (2-12)**
specifies the 24-bit addressing-mode address for the parameter list.

COMPLETION CODES

When the system returns control to your problem program, the low-order byte of register 15 contains a return code; the low-order byte of register 0 contains a reason code:

Return Code (15)	Reason Code (0)	Meaning
00 (X'00')		Successful completion. Register 0 contains the number of data blocks in the control unit buffer.
04 (X'04')	01 (X'01')	Incorrect parameter.
04 (X'04')	02 (X'02')	Incorrect DCB or a DEBCHK error.
04 (X'04')	03 (X'03')	Environmental error.
04 (X'04')	04 (X'04')	Incorrect input to NOTE.
04 (X'04')	05 (X'05')	Device does not support buffering.
04 (X'04')	11 (X'0B')	Unsuccessful call to ESTAE macro.
04 (X'04')	12 (X'0C')	Unsuccessful GETMAIN request.
08 (X'08')		Permanent I/O error during read block ID or synchronize command.
12 (X'0C')		Permanent I/O error on the last channel program with loss of data. **Note:** If you specified a SYNAD option in the DCB and issue a PUT or CHECK macro after this error occurs, your program cannot enter the SYNAD routine.

TRUNC—TRUNCATE AN OUTPUT BUFFER (QSAM OUTPUT—FIXED- OR VARIABLE-LENGTH BLOCKED RECORDS)

The TRUNC macro instruction causes the current output buffer to be regarded as full. The next PUT or PUTX macro instruction specifying the same data control block uses the next buffer to hold the logical record.

When a variable-length spanned record is truncated and logical record interface, or extended logical record interface, is specified (that is, if BFTEK=A is specified in the DCB macro instruction, or if a BUILDRCD macro instruction is issued, or if DCBLRECL=0K or nnnnnK is specified), the system segments and writes the record before truncating the buffer. Therefore, the block being truncated is the one that contains the last segment of the spanned record.

The TRUNC macro instruction is ignored if it is used for unblocked records, if it is used when a buffer is full, or if it is used without an intervening PUT or PUTX macro instruction.

The TRUNC macro is written:

[symbol]	**TRUNC**	dcb address

dcb address—RX-Type Address, (2-12), or (1)
: The dcb address operand specifies the address of the data control block for the sequential data set opened for output. The record format in the data control block must not indicate standard blocked records (RECFM=FBS).

WAIT—WAIT FOR ONE OR MORE EVENTS (BDAM, BISAM, BPAM, AND BSAM)

The WAIT macro instruction is used to inform the control program that performance of the active task cannot continue until one or more specific events, each represented by a different ECB (event control block), have occurred. In the context of this manual, the ECBs represent completion of I/O processing associated with a READ or WRITE macro. ECBs are located at the beginning of access method DECBs (data event control blocks), so that the DECB name provided in READ and WRITE macros is also used for WAIT. (A description of the ECB is found in Appendix A, "Status Information Following an Input/Output Operation" on page 192. For information on when to use the WAIT macro, see _Data Administration Guide_.)

The control program takes the following action:

- For each event that has already occurred (each ECB is already posted), the count of the number of events is decreased by 1.
- If the number of events is 0 by the time the last event control block is checked, control is returned to the instruction following the WAIT macro instruction.
- If the number of events is not 0 by the time the last ECB is checked, control is not returned to the issuing program until sufficient ECBs are posted to bring the number to 0. Control is then returned to the instruction following the WAIT macro instruction.
- The events will be posted complete by the system when all I/O has been completed, temporary errors have been corrected, and length checking has been performed. The DECB is not checked for errors or exceptional conditions, nor are end-of-volume procedures initiated. Your program must perform these operations.

The WAIT macro is written:

[_symbol_]	WAIT	[_number of events_] {,ECB=_addr_|ECBLIST=_addr_} [,LONG={YES|_NO_}]

number of events
specifies a decimal integer from 0 to 255. Zero is an effective NOP instruction; 1 is assumed if the operand is omitted. The number of events must not exceed the number of event control blocks. You may also use register notation (2-12).

ECB=_addr_
specifies the address of the event control block (or DECB) representing the single event that must occur before processing can continue. The operand is valid only if the number of events is specified as 1 or is omitted.

addr
specify RX type or use register notation (1-12).

ECBLIST=_addr_
specifies the address of a virtual storage area containing one or more consecutive fullwords on a fullword boundary. Each fullword contains the address of an event control block (or DECB); the high-order bit in the last word (address) must be set to 1 to indicate the end of the list. The number of event control blocks must be equal to or greater than the specified number of events.

LONG=[YES|<u>NO</u>]
specifies whether the task is entering a long wait or a regular wait. Normally, I/O events should not be considered 'long' unless it is anticipated that operator intervention will be required.

Caution: A job step with all its tasks in a WAIT condition terminates upon expiration of the time limits that apply to it.

Access method ECBs are maintained entirely by the access methods and supporting control program facilities. The user may inspect access method ECBs, but should never modify them.

WRITE—WRITE A BLOCK (BDAM)

The WRITE macro instruction causes the system to add or replace a block in an existing direct data set. (This version of the WRITE macro instruction cannot be used to create a direct data set because no capacity record facilities are provided.) Control may be returned before the block is written. The output operation must be tested for completion using a CHECK or WAIT macro instruction. A data event control block, shown in Appendix A, "Status Information Following an Input/Output Operation" on page 192, is constructed as part of the macro expansion.

The standard form of the WRITE macro instruction is written as follows (the list and execute forms are shown following the descriptions of the standard form):

[symbol]	WRITE	decb name ,type ,dcb address ,{area address|'S'} ,{length|'S'} ,{key address|'S'|0} ,block address

decb name—symbol
specifies the name assigned to the data event control block created as part of the macro expansion.

type—{DA[F]}
{DI[F|X]}
{DK[F|X]}
is coded in one of the combinations shown to specify the type of write operation and optional services performed by the system:

DA
specifies that a new data block is to be added to the data set in the first available space; the search for available space starts at the device address indicated in the area specified in the block address operand. Fixed-length records (with keys only) are added to a data set by replacing dummy records. Variable-length records (with or without keys) are added to a data set by using available space on a track. (For more information on adding records to a direct data set, see Data Administration Guide. The description of the DCB macro instruction, LIMCT operand, contains a description of the search.)

DI
specifies that a data block and key, if any, are to be written at the device address indicated in the area specified in the block address operand. Any attempt to write a capacity record (R0) is an invalid request when relative track addressing or actual device addressing are used, but when relative block addressing is used, relative block 0 is the first data block in the data set.

DK
specifies that a data block (only) is to be written using the key in the area specified by the key address operand as a search argument; the search for the block starts at the device address indicated in the area specified in the block address operand. The description of the DCB macro instruction, LIMCT operand, contains a description of the search.

F

requests that the system provide block position feedback into the area specified in the block address operand. This character can be coded as a suffix to **DA**, **DI**, or **DK** as shown above.

X

requests that the system release the exclusive control requested by a previous READ macro instruction and provide block position feedback into the area specified in the block address operand. This character can be coded as a suffix to **DI** or **DK** as shown above.

dcb address—A-Type Address or (2-12)
specifies the address of the data control block for the opened BDAM data set.

area address—A-Type Address, (2-12), or **'S'**
specifies the address of the area that contains the data block to be written. **'S'** can be coded instead of an area address only if the data block (or key and data) are contained in a buffer provided by dynamic buffering; that is, **'S'** was coded in the area address operand of the associated READ macro instruction. If **'S'** is coded in the WRITE macro instruction, the area address from the READ macro instruction data event control block must be moved into the WRITE macro instruction data event control block; the buffer area acquired by dynamic buffering is released after the WRITE macro instruction is executed. For a description of the data event control block, see Appendix A, "Status Information Following an Input/Output Operation" on page 192.

length—symbol, decimal digit, absexp, (2-12) or **'S'**
specifies the number of data bytes to be written up to a maximum of 32760. If **'S'** is coded, it specifies that the system uses the value in the block size (DCBBLKSI) field as the length. When undefined-length records are used, if the WRITE macro instruction is for update and the length specified differs from the original block, the new block will be truncated or padded with binary zeros accordingly. The problem program can check for this situation in the SYNAD routine.

If the length operand is omitted for format-U records, no error indication is given when the program is assembled, but the problem program must insert a length into the data event control block before the WRITE macro instruction is executed.

key address—A-Type Address, (2-12), **'S'**, or **0**
specifies the address of the area that contains the key to be used. **'S'** is specified instead of an address only if the key is contained in an area acquired by dynamic buffering. If the key is not written or used as a search argument, zero is specified instead of a key address.

block address—A-Type Address or (2-12)
specifies the address of the area that contains the relative block address, relative track address, or actual device address used in the output operation. The length of the area depends on the type of addressing used and if the feedback option (OPTCD=F) is specified in the data control block.

If OPTCD=F has been specified in the DCB macro and F or X is specified in the WRITE macro, you must provide a relative block address in the form specified by OPTCD in the DCB macro. For example, if OPTCD=R is specified, you must provide a 3-byte relative block address; if OPTCD=A is specified, you must provide an 8-byte actual device address (MBBCCHHR); if neither is specified, you must provide a 3-byte relative address (TTR).

If OPTCD=F has not been specified in the DCB macro and F or X is specified in the WRITE macro, then you must provide an 8-byte actual device address (MBBCCHHR) even if relative block or relative track addressing is being used.

WRITE—WRITE A LOGICAL RECORD OR BLOCK OF RECORDS (BISAM)

The WRITE macro instruction causes the system to add or replace a record or replace an updated block in an existing indexed sequential data set. Control may be returned to the problem program before the block or record is written. The output operation must be tested for completion using a WAIT or CHECK macro instruction. A data event control block, shown in Appendix A, "Status Information Following an Input/Output Operation" on page 192, is constructed as part of the macro expansion.

The standard form of the WRITE macro instruction is written as follows (the list and execute forms are shown following the descriptions of the standard form):

[symbol]	WRITE	decb name ,type ,dcb address ,{area address|'S'} ,{length|'S'} ,key address

decb name—symbol
: specifies the name assigned to the data event control block created as part of the macro expansion.

type—{K|KN}
: is coded as shown to specify the type of write operation:

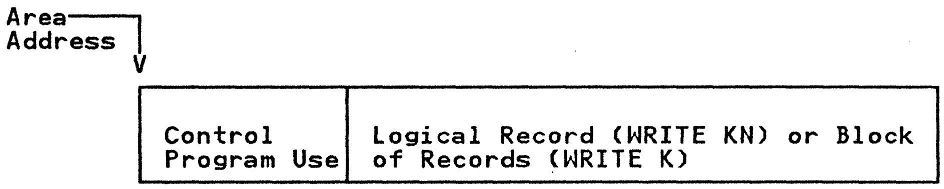
K
: specifies that either an updated unblocked record or a block containing an updated record is to be written. If the record has been read using a READ KU macro instruction, the data event control block for the READ macro instruction must be used as the data event control block for the WRITE macro instruction, using the execute form of the WRITE macro instruction.

KN
: specifies that a new record is to be written, or a variable-length record is to be rewritten with a different length. All records or blocks of records read using READ KU macro instructions for the same data control block must be written back before a new record can be added, except when the READ KU and WRITE KN reference the same DECB.

dcb address—A-Type Address or (2-12)
: specifies the address of the data control block for the opened existing indexed sequential data set. If a block is written, the data control block address must be the same as the dcb address operand in the corresponding READ macro instruction.

area address—A-Type Address, (2-12), or 'S'
: specifies the address of the area containing the logical record or block of records to be written. The first 16 bytes of this area are used by the system and should not contain your data. The area address must specify a different area than the key address. When new records are written (or when variable-length records are rewritten with a different length), the area address of the new record must always be supplied by the problem program. This area may be altered by the system. 'S' may be coded instead of an address only if the block of records is contained in an area provided by dynamic buffering; that is, 'S' was coded for the area address operand in the associated READ KU macro instruction. This area is released after execution of a WRITE macro instruction using the same DECB. The area can also be released by a FREEDBUF macro instruction.

The following illustration shows the format of the area:



Indexed sequential buffer and work area requirements are discussed in _Data Administration Guide_.

length—symbol, decimal digit, absexp, (2-12) or 'S'
specifies the number of data bytes to be written, up to a maximum of 32760. Specify 'S' unless a variable-length record will be rewritten with a different length.

key address—A-Type Address or (2-12)
specifies the address of the area containing the key of the new or updated record. The key address must specify a different area than the area address. For blocked records, this is not necessarily the high key in the block. For unblocked records, this field should not overlap with the work area specified in the MSWA parameter of the DCB macro instruction.

Note: When new records are written, the key area may be altered by the system.

WRITE—WRITE A BLOCK (BPAM AND BSAM)

The WRITE macro instruction causes the system to add or replace a block in a sequential or partitioned data set being created or updated. Control may be returned to the problem program before the block is written. The output operation must be tested for completion using the CHECK macro instruction. A data event control block, shown in Appendix A, "Status Information Following an Input/Output Operation" on page 192, is constructed as part of the macro expansion.

If translation from EBCDIC code to ISCII/ASCII code is requested, issuing multiple WRITE macro instructions for the same record causes an error because the first WRITE macro instruction issued causes the output data in the output buffer to be translated into ISCII/ASCII code.

If the OPEN macro instruction specifies UPDAT, both the READ and WRITE macro instructions must reference the same data event control block. See the list form of the READ or WRITE macro instruction for a description of how to construct a data event control block; see the execute form of the READ or WRITE macro instruction for a description of modifying an existing data event control block.

The standard form of the WRITE macro instruction is written as follows (the list and execute forms are shown following the descriptions of the standard form):

[symbol]	**WRITE**	decb name ,SF ,dcb address ,area address [,length|,'S']

decb name—symbol
    specifies the name assigned to the data event control block created as part of the macro expansion.

SF
    specifies normal, sequential, forward operation.

dcb address—A-Type Address, or (2-12)
    specifies the address of the data control block for the opened data set being created or processed. If the data set is being updated, the data control block address must be the same as the dcb address operand in the corresponding READ macro instruction.

area address—A-Type Address or (2-12)
    specifies the address of the area that contains the data block to be written; if a key is written, the key must precede the data in the same area.

length—symbol, decimal digit, absexp, (2-12) or 'S'
    specifies the number of bytes to be written; this operand is specified for only undefined-length records (RECFM=U) or for ASCII records (RECFM=D) when the DCB BUFOFF operand is zero. For AL tapes, the maximum length is 2048 bytes; otherwise, the maximum length is 32760 bytes. 'S' can be coded to indicate that the value specified in the block size (DCBBLKSI) field of the data control block is used as the length to be written. The length operand should be omitted for all record formats except format-U and format-D (when BUFOFF=0).

    If the length operand is omitted for format-U or format-D (with BUFOFF=0) records, no error indication is given when the program is assembled, but the problem program must insert a length into the data event control block before the WRITE macro is issued.

WRITE—WRITE A BLOCK (CREATE A BDAM DATA SET WITH BSAM)

The WRITE macro instruction causes the system to add a block to the direct data set being created. For fixed-length blocks, the system writes the capacity record automatically when the current track is filled; for variable and undefined-length blocks, a WRITE macro instruction must be issued for the capacity record. Control may be returned before the block is written. The output operations must be tested for completion using a CHECK macro instruction. A data event control block, shown in Appendix A, "Status Information Following an Input/Output Operation" on page 192, is constructed as part of the macro expansion.

The standard form of the WRITE macro instruction is written as follows (the list and execute forms are shown following the descriptions of the standard form):

[symbol]	WRITE	decb name ,type ,dcb address ,area address [,length|,'S'] [,next address]

decb name—symbol
: specifies the name assigned to the data event control block created as part of the macro expansion.

type—{SF|SFR|SD|SZ}
: is coded as shown, to specify the type of write operation performed by the system:

SF
: specifies that a new data block is to be written in the data set.

SFR
: specifies that a new variable-length spanned record is to be written in the data set, and next address feedback is requested. This operand can be specified only for variable-length spanned records (BFTEK=R and RECFM=VS are specified in the data set control block). If type SFR is specified, the next address operand must be included.

SD
: specifies that a dummy data block is to be written in the data set; dummy data blocks can be written only when fixed-length records with keys are used.

SZ
: specifies that a capacity record (R0) is to be written in the data set; capacity records can be written only when variable-length or undefined-length records are used.

dcb address—A-Type Address or (2-12)
: specifies the address of the data control block opened for the data set being created. DSORG=PS (or PSU) and MACRF=WL must be specified in the DCB macro instruction to create a BDAM data set.

area address—A-Type Address or (2-12)
: specifies the address of the area that contains the data block to be added to the data set. If keys are used, the key must precede the data in the same area. For writing capacity records (SZ), the area address is ignored and can be omitted (the system supplies the information for the capacity record). For writing dummy data blocks (SD), the area need be only large enough to hold the key plus one data byte. The system constructs a dummy key with the

first byte set to all 1 bits (hexadecimal FF) and adds the block number in the first byte following the key. When a dummy block is written, a complete block is written from the area immediately following the area address; therefore, the area address plus the value specified in the BLKSIZE and KEYLEN operands must be within the area allocated to the program writing the dummy blocks.

length—symbol, decimal digit, absexp, (2-12), or 'S'
: is used only when undefined-length (RECFM=U) blocks are being written. The operand specifies the length of the block, in bytes, up to a maximum of 32760. If 'S' is coded, it specifies that the system is to use the length in the block size (DCBBLKSI) field of the data control block as the length of the block to be written.

  If the length operand is omitted for format-U records, no error indication is given when the program is assembled, but the problem program must insert a length into the data event control block before the WRITE is issued.

next address—A-Type Address or (2-12)
: specifies the address of the area where the system places the relative track address of the next record to be written. Next address feedback can be requested only when variable-length spanned records are used.

Note: When variable-length spanned records are used (RECFM=VS and BFTEK=R are specified in the data control block), the system writes capacity records (R0) automatically in the following cases:

- When a record spans a track.
- When the record cannot be written completely on the current volume. In this case, all capacity records of remaining tracks on the current volume are written; tracks not written for this reason are still counted in the search limit specified in the LIMCT operand of the data control block.
- When the record written is the last record on the track, the remaining space on the track cannot hold more than eight bytes of data.

COMPLETION CODES FOR WRITE—WRITE A BLOCK (CREATE A BDAM DATA SET WITH BSAM)

After the write has been scheduled and control has been returned to the user's program, the three high-order bytes of register 15 are set to 0; the low-order byte contains one of the following return codes:

Return Code	Meaning		
	Fixed-Length	Variable or Undefined-Length	
	(SF or SD)	(SF or SFR)	(SZ)
00	Block will be written. (If the previous return code was 08, a block is written only if the DD statement specifies secondary space allocation and sufficient space is available.)	Block will be written. (If the previous return code was 08, a block is written only if the DD statement specifies secondary space allocation and sufficient space is available.)	Capacity record was written; another track is available.
04	Block will be written, followed by a capacity record. (If the previous return code was 08, a block is written only if the DD statement specifies secondary space allocation and sufficient space is available.)	Block was not written; write a capacity record (SZ) to describe the current track, then reissue the WRITE macro instruction.	—
08	Block will be written, followed by a capacity record. The next block requires secondary space allocation.	—	Capacity record was written. The next block requires secondary space allocation. This code is not issued if the WRITE SZ is the only WRITE macro instruction issued on a one-track secondary extent.
0C	Block will not be written; issue a CHECK macro instruction for the previous WRITE macro instruction, then reissue the WRITE macro instruction.	Block will not be written; issue a CHECK macro instruction for the previous WRITE macro instruction, then reissue the WRITE macro instruction.	Block will not be written; issue a CHECK macro instruction for the previous WRITE macro instruction, then reissue the WRITE macro instruction.

Note: For fixed-length records, the return codes are unpredictable when writing one record per track with one track per extent.

WRITE—LIST FORM

The list form of the WRITE macro instruction is used to construct a data management parameter list in the form of a data event control block (DECB). For a description of the various fields in the DECB for each access method, see Appendix A, "Status Information Following an Input/Output Operation" on page 192.

The description of the standard form of the WRITE macro instruction provides the explanation of the function of each operand. The description of the standard form also indicates the operands used for each access method and the meaning of 'S' when coded for the area address, length, and key address operands. For each access method, 'S' can be coded only for those operands for which it can be coded in the standard form of the macro instruction. The format description below indicates the optional and required operands in the list form only, but does not indicate optional and required operands for any specific access method.

The list form of the WRITE macro is written:

```
[symbol]   WRITE     decb name
                     ,type
                     ,[dcb address]
                     ,[area address|'S']
                     ,[length|'S']
                     ,[key address|'S']
                     ,[block address]
                     ,[next address]
                     ,MF=L
```

decb name—symbol

type—Code one of the types shown in the standard form

dcb address—A-Type Address

area address—A-Type Address or 'S'

length—symbol, decimal digit, absexp, or 'S'

key address—A-Type Address or 'S'

block address—A-Type Address

next address—A-Type Address

MF=L—Coded as shown
specifies that the WRITE macro instruction is used to create a data event control block that will be referenced by an execute-form instruction.

WRITE—EXECUTE FORM

A remote data management parameter list (data event control block) is used in, and can be modified by, the execute form of the WRITE macro instruction. The data event control block can be generated by the list form of either a READ or WRITE macro instruction.

The description of the standard form of the WRITE macro instruction provides the explanation of the function of each operand. The description of the standard form also indicates the operands used for each access method and the meaning of **'S'** when coded for the area address, length, and key address operands. For each access method, **'S'** can be coded only for those operands for which it can be coded in the standard form of the macro instruction. The format description below indicates the optional and required operands in the execute form only, but does not indicate the optional and required operands for any specific access method.

The execute form of the WRITE macro instruction is written as follows:

[symbol]	**WRITE**	decb address ,type ,[dcb address] ,[area address|'S'] ,[length|'S'] ,[key address|'S'] ,[block address] ,[next address] ,MF=E

decb address—RX-Type Address or (2-12)

type—Code one of the types shown in the standard form

dcb address—RX-Type Address or (2-12)

area address—RX-Type Address, (2-12), or **'S'**

length—symbol, decimal digit, absexp, (2-12), or **'S'**

key address—RX-Type Address, (2-12), or **'S'**

block address—RX-Type Address or (2-12)

next address—RX-Type Address or (2-12)

MF=E—Coded as shown
specifies that the execute form of the WRITE macro instruction is used, and an existing data event control block (specified in the decb address operand) is to be used by the access method.

XLATE—TRANSLATE TO AND FROM ISCII/ASCII (BSAM AND QSAM)

The XLATE macro instruction is used to translate the data in an area in virtual storage from ISCII/ASCII code to EBCDIC code or from EBCDIC code to ISCII/ASCII code.

To determine the ISCII/ASCII to EBCDIC or EBCDIC to ISCII/ASCII translation codes, see ANSI X3.26-1980 (American National Standard for Hollerith punch card code). When translating EBCDIC code to ISCII/ASCII code, all ISCII/ASCII code not having an EBCDIC equivalent is translated to X'3F'. When translating ISCII/ASCII code to EBCDIC code, all EBCDIC code not having an ISCII/ASCII equivalent is translated to X'1A'. Because Version 3 ISCII/ASCII uses only 7 bits in each byte, bit 0 is always set to 0 during EBCDIC to ISCII/ASCII translation and is expected to be 0 during ISCII/ASCII to EBCDIC translation.

The XLATE macro is written:

[symbol]	**XLATE**	area address ,length **[,TO={A|E}]**

area address—RX-Type Address, symbol, decimal digit, absexp, (2-12), or (1)
specifies the address of the area that is to be translated.

length—symbol, decimal digit, absexp, (2-12), or (0)
specifies the number of bytes to be translated.

TO={A|E}
specifies the type of translation requested. If this operand is omitted, **E** is assumed. The following describes the characters that can be specified:

A
specifies that translation from EBCDIC code to ISCII/ASCII code is requested.

E
specifies that translation from ISCII/ASCII code to EBCDIC code is requested.

APPENDIX A. STATUS INFORMATION FOLLOWING AN INPUT/OUTPUT OPERATION

Following an input/output operation, the control program makes certain status information available to the problem program. This information is a 2-byte exception code, or a 16-byte field of standard status indicators, or both.

Exception codes are provided in the data control block (QISAM), or in the data event control block (BISAM and BDAM). The data event control block is described below, and the exception code lies within the block as shown in the illustration for the data event control block. If a DCBD macro instruction is coded, the exception code in a data control block can be addressed as two 1-byte fields, DCBEXCD1 and DCBEXCD2. For more information, see Data Facility Product: Customization.

Status indicators are available only to the error analysis routine designated by the SYNAD entry in the data control block. A pointer to the status indicators is provided either in the data event control block (BSAM, BPAM, and BDAM), or in register 0 (QISAM and QSAM). For more information, see Data Facility Product: Customization.

DATA EVENT CONTROL BLOCK

A data event control block is constructed as part of the expansion of READ and WRITE macro instructions and is used to pass parameters to the control program, help control the read or write operation, and receive indications of the success or failure of the operation. The data event control block is named by the READ or WRITE macro instruction, begins on a fullword boundary, and contains the information shown in the following illustration:

	Field Contents		
Offset from DECB Address (Bytes)	**BSAM and BPAM**	**BISAM**	**BDAM**
0	ECB	ECB	ECB[1]
+4	Type	Type	Type
+6	Length	Length	Length
+8	DCB address	DCB address	DCB address
+12	Area address	Area address	Area address
+16	IOB address	Logical record address	IOB address
+20		Key address	Key address
+24		Exception code (2 bytes)	Block address
+28			Next address

[1] The control program returns exception codes in bytes +1 and +2 of the ECB.

APPENDIX B. DATA MANAGEMENT MACRO INSTRUCTIONS AVAILABLE BY ACCESS METHOD

Macro Instruction	BDAM	BISAM	BPAM	BSAM	QISAM	QSAM
BLDL			X			
BSP				X		
BUILD	X	X	X	X	X	X
BUILDRCD						X
CHECK	X	X	X	X		
CHKPT	X	X	X	X	X	X
CLOSE	X	X	X	X	X	X
CNTRL				X		X
DCB	X	X	X	X	X	X
DCBD	X	X	X	X	X	X
ESETL					X	
FEOV				X		X
FIND			X			
FREEBUF	X	X	X	X		
FREEDBUF	X	X				
FREEPOOL	X	X	X	X	X	X
GET					X	X
GETBUF	X	X	X	X		
GETPOOL	X	X	X	X	X	X
MSGDISP	X	X	X	X	X	X
NOTE			X	X		
OPEN	X	X	X	X	X	X
PDAB						X
PDABD						X
POINT			X	X		
PRTOV				X		X
PUT					X	X
PUTX					X	X
READ	X	X	X	X		
RELEX	X					
RELSE					X	X
SETL					X	
SETPRT				X		X
STOW			X			
SYNADAF	X	X	X	X	X	X
SYNADRLS	X	X	X	X	X	X
SYNCDEV	X	X	X	X	X	X
TRUNC						X
WAIT	X	X	X	X		
WRITE	X	X	X	X		
XLATE				X		X

APPENDIX C. DEVICE CAPACITIES

The following information provides a guide to coding the block size (BLKSIZE) and logical record length (LRECL) operands in the DCB macro instruction. These values can be used to determine the maximum block size and logical record length for a given device, and they can be used to determine the optimum blocking factor when records are to be blocked.

CARD READERS AND CARD PUNCHES

Format F, V, or U records are accepted by readers and punches, but the logical record length for a card reader or card punch is fixed at 80 bytes. If the optional control character is specified, the logical record length is 81 (the control character is not part of the data record). If card image mode is used, the buffer required to contain the data must be 160 bytes.

PRINTERS

The following table shows the record length that can be specified for the various printers. In some cases, two values are shown; except for the 3800, the larger of the two values requires that an optional feature be installed on the printer being used. If the optional control character is specified to control spacing and skipping, the record length is specified as one greater than the actual data length (the control character is not part of the data record).

Printer	Record Length
1403 Printer	120 or 132 bytes
3203 Printer	132 bytes
3211 Printer	132 or 150 bytes
3525 Card Punch, Print Feature	64 bytes
3800 Printing Subsystem	136 bytes for 10 pitch 163 bytes for 12 pitch 204 bytes for 15 pitch

MAGNETIC TAPE UNITS

3480 Magnetic Tape Subsystem (18 track)	32760 bytes

DIRECT ACCESS DEVICES

The following table shows the capacity of direct access devices by track, cylinder, and total capacity in bytes.

Device	Maximum Block Size/Track (Note 1)	Tracks per Cylinder	Number of Cylinders (Notes 1 and 2)	Track Capacity (Bytes)
2305-2	14660	8	96	14858[6]
3330/3333 (Model 1)[3]	13030	19	404	13165[6]
3330/3333 (Model 11)	13030	19	808	13165[6]
3340/3344[4]	8368	12	696	8535[6]
3350	19069	30	555	19254[6]
3375	32760[5]	12	959	36000[6]
3380 (Models A04, AA4, and B04)	32760[5]	15	885	47968[6]
3380 (Models AD4 and BD4)	32760[5]	15	885	47968[6]
3380 (Models AE4 and BE4)	32760[5]	15	1770	47968[6]

1 Capacity indicated in bytes (when R0 is used by the IBM programming system).

2 Excludes alternate cylinders.

3 The Mass Storage System (MSS) virtual volumes assume the characteristics of the 3330/3333, Model 1.

4 The 3344 is functionally equivalent to the 3340 Model 70.

5 The largest record that can be written on a track for the 3375 is 35616 and for all 3380 models is 47476. However, for these devices the largest block size supported by the standard access methods is 32760.

6 This value is different from the maximum block size per track because each block on the track includes an overhead for this device.

Each record written on a direct access device requires some 'device overhead.' The term device overhead means the space required by the device for address markers, count areas, gaps between the count, key, and data areas, and gaps between blocks. The following calculations can be used to compute the number of bytes required for each data block including the space required for device overhead. Note that any fraction of a byte must be ignored. For example, if the calculation results in 15.644 bytes, 15 bytes must be used to determine track capacity.

Device	Blocks With Keys	Blocks Without Keys
2305-2	289+KL[1]+DL[2]	198+DL
3330/3333 (Model 1 or 11)[3]	191+KL+DL	135+DL
3340/3344	242+KL+DL	167+DL
3350	267+KL+DL	185+DL
3375	224+((KL+191)/32)(32)+((DL+191)/32)(32)	224+((DL+191)/32)(32)
3380 (Models A04, AA4, and B04)	256+((KL+267)/32)(32)+((DL+267)/32)(32)	256+((DL+267)/32)(32)
3380 (Models AD4 and BD4)	256+((KL+267)/32)(32)+((DL+267)/32)(32)	256+((DL+267)/32)(32)
3380 (Models AE4 and BE4)	256+((KL+267)/32)(32)+((DL+267)/32)(32)	256+((DL+267)/32)(32)

[1] KL is key length.

[2] DL is data length.

[3] The Mass Storage System (MSS) virtual volumes assume the characteristics of the 3330/3333, Model 1.

When track overflow is used or variable-length spanned records are written, the size of a data block or logical record can exceed the capacity of a single track on the direct access device used.

APPENDIX D. DCB EXIT LIST FORMAT AND CONTENTS

The following shows the format and contents that must be supplied by the problem program when the EXLST operand is specified in a DCB macro instruction. The exit list must begin on a fullword boundary and each entry in the list requires one fullword.

Entry Type	Hex Code	3-Byte Address—Purpose
Inactive entry	00	Ignore the entry; it is not active.
Input header label exit	01	Process a user input header label.
Output header label exit	02	Create a user output header label.
Input trailer label exit	03	Process a user input trailer label.
Output trailer label exit	04	Create a user output trailer label.
Data control block exit	05	Take a data control block exit.
End-of-volume exit	06	Take an end-of-volume exit.
JFCB exit	07	JFCB address for RDJFCB and OPEN TYPE=J SVCs.
	08	Reserved.
I/O error processing exit	09	User option to process I/O errors.
User totaling area	0A	Address of beginning of user's totaling area.
Block count exit	0B	Take a block-count-unequal exit.
Defer input trailer label	0C	Defer processing of a user input trailer label from end-of-data until closing.
Defer nonstandard input trailer label	0D	Defer processing a nonstandard input trailer label on magnetic tape unit from end-of-data until closing (no exit routine address).
	0E-0F	Reserved.
FCB image	10	Define an FCB image.
DCB abend exit	11	Examine the abend condition and select one of several options.
QSAM parallel input	12	Address of the PDAB for which this DCB is a member.
| Allocation retrieval list	13	Retrieve allocation information for one or more data sets with RDJFCB.
	14	Reserved.
JFCBE exit	15	Take an exit during OPEN to allow user to examine JCL=specified setup requirements for a 3800 printer.
	16	Reserved.

Entry Type	Hex Code	3-Byte Address—Purpose
OPEN/EOV nonspecific tape volume mount	17	Option to specify a tape volume serial number.
OPEN/EOV volume security/verification	18	Verify a tape volume and some security checks.
	19-7F	Reserved.
Last entry	80	Treat this entry as the last entry in the list. This code can be specified with any of the above but must always be specified with the last entry.

The list can be dynamically shortened during execution by setting the high-order bit of the word to a value of 1. An entry in the list can be made inactive dynamically by setting the high-order byte of the word to a value of hexadecimal 00.

When control is passed to an exit routine, the general registers contain the following information:

Register	Contents
0	Variable; the contents depend on the exit routine used.
1	The three low-order bytes contain either the address of the DCB currently being processed or, when certain exits are taken, the address of the exit parameter list. These exits are: user-label exits (X'01'-'04'), deferred nonstandard input trailer exit (X'0D'), and DCB abend exit (X'11').
2-13	Contents prior to execution of the macro instruction.
14	Return address (must not be altered by the exit routine).
15	Address of the exit routine entry point.

The conventions for saving and restoring registers are as follows:

- The exit routine must preserve the contents of register 14. It need not preserve the contents of other registers. The control program restores registers 2 through 13 before returning control to the problem program.

- The exit routine must not use the save area whose address is in register 13, because this area is used by the control program. If the exit routine calls another routine or issues supervisor or data management macro instructions, it must provide the address of a new save area in register 13.

For a detailed description of each exit list processing option, see Data Administration Guide.

APPENDIX E. CONTROL CHARACTERS

Each logical record, in all record formats, can contain an optional control character. This control character is used to control stacker selection on a card punch or card read punch, or it is used to control printer spacing and skipping. If a record containing an optional control character is directed to any other device, it is considered to be the first data byte, and it does not cause a control function to occur.

In format-F and format-U records, the optional control character must be in the first byte of the logical record.

In format-V or format-D records, the optional control character must be in the fifth byte of the logical record, immediately following the record descriptor word of the record.

Two control character options are available. A control character option is selected by coding the appropriate character in the RECFM operand of the DCB macro instruction. If either option is specified in the data control block, a control character must be included in each record, and other spacing or stacker selection options also specified in the data control block are ignored.

MACHINE CODE

The record format field in the data control block indicates that the machine code control character has been placed in each logical record. If the record is written, the appropriate byte must contain the command code bit configuration specifying both the write and the desired carriage or stacker select operation.

The machine code control characters for a printer are:

Print—Then Act	Action	Act Immediately Without Printing
X'01'	Print only (no space)	
X'09'	Space 1 line	X'0B'
X'11'	Space 2 lines	X'13'
X'19'	Space 3 lines	X'1B'
X'89'	Skip to channel 1	X'8B'
X'91'	Skip to channel 2	X'93'
X'99'	Skip to channel 3	X'9B'
X'A1'	Skip to channel 4	X'A3'
X'A9'	Skip to channel 5	X'AB'
X'B1'	Skip to channel 6	X'B3'
X'B9'	Skip to channel 7	X'BB'
X'C1'	Skip to channel 8	X'C3'
X'C9'	Skip to channel 9	X'CB'
X'D1'	Skip to channel 10	X'D3'
X'D9'	Skip to channel 11	X'DB'
X'E1'	Skip to channel 12	X'E3'

The machine code control characters for a card read punch device are as follows:

Control Code	Action
X'01'	Select stacker 1
X'41'	Select stacker 2
X'5A'[1]	Change from line mode to page mode
X'81'	Select stacker 3

[1] The 3800 Model 3 all-points-addressable mode uses this code to change from compatibility to page mode.

Other command codes for specific devices are contained in IBM System Reference Library publications describing the control units or devices.

ISO/ANSI/FIPS CONTROL CHARACTERS

In place of machine code, control characters defined by the International Organization for Standardization (ISO), American National Standards Institute (ANSI), or the Federal Information Processing Standards (FIPS) can be specified. These characters must be represented in EBCDIC code.

International Organization for Standardization (ISO), American National Standards Institute (ANSI), or Federal Information Processing Standards (FIPS) control characters are as follows:

Code	Action before Printing a Line
b	Space one line (blank code)
0	Space two lines
-	Space three lines
+	Suppress space
1	Skip to channel 1
2	Skip to channel 2
3	Skip to channel 3
4	Skip to channel 4
5	Skip to channel 5
6	Skip to channel 6
7	Skip to channel 7
8	Skip to channel 8
9	Skip to channel 9
A	Skip to channel 10
B	Skip to channel 11
C	Skip to channel 12
Code	**Action after Punching a Card**
V	Select punch pocket 1
W	Select punch pocket 2
X'5A'[1]	Change from line mode to page mode

[1] The 3800 Model 3 all-points-addressable mode uses this code to change from compatibility to page mode.

These control characters include those defined by ANSI FORTRAN. If any other character is specified, it is interpreted as 'b' or V, depending on the device being used; no error indication is returned.

APPENDIX F. DATA CONTROL BLOCK SYMBOLIC FIELD NAMES

The following describes data control block fields that contain information that defines the data characteristics and device requirements for a data set. Each of the fields described shows the values that result from specifying various options in the DCB macro instruction. These fields can be referred to by the problem program through the use of a DCBD macro instruction that creates a dummy control section (DSECT) for the data control block. Fields that contain addresses are 4 bytes long and are aligned on a fullword boundary. If the problem program inserts an address into a field, the address must be inserted into the low-order 3 bytes of the field without changing the high-order byte.

The contents of some fields in the data control block depend on the device and access method being used. A separate description is provided when the contents of the field are not common to all device types and access methods.

DATA CONTROL BLOCK—COMMON FIELDS

Offset	Bytes and Alignment	Field Name	Description
26(1A)	2	DCBDSORG	Data set organization.
			Code
		1...	IS Indexed sequential.
		.1..	PS Physical sequential.
		..1.	DA Direct organization.
		...x xx..	Reserved bits.
		.... ..1.	PO Partitioned organization.
		1	U Unmovable—the data set contains location-dependent information.
40(28)	8	DCBDDNAM	Eight-byte name of the data definition statement that defines the data set associated with this DCB. (Before DCB is opened.)
40(28)	2	DCBTIOT	(After DCB is opened.) Offset from the TIOT origin to the TIOELNGH field in the TIOT entry for the DD statement associated with this DCB.
42(2A)	2	DCBMACRF	This field may only be referenced during and after OPEN. It is common to all uses of the DCB and is created by moving the DCBMACR field into this area.
45(2D)	.3	DCBDEBA	(After DCB is opened.) Address of the associated DEB.
48(30)	1	DCBOFLGS	Flags used by open routine.
		...1	OPEN has completed successfully.
		.... 1...	Set to 1 by problem program to indicate concatenation of unlike attributes.
		.... ..0.	Set to 0 by an I/O support function when that function takes a user exit. It is set to 0 to inhibit other I/O support functions from processing this DCB.
		.... ..1.	Set to 1 on return from the user exit to the I/O support function that took the exit.
50(32)	..2	DCBMACR (Before OPEN)	Macro instruction reference before OPEN. Major macro instructions and various options associated with them. Used by the open routine to determine access method. Used by the access method executes in conjunction with other parameters to determine which load modules are required. This field is moved to overlay part of DCBDDNAM at open time and becomes the DCBMACRF field. This field is common to all uses of the DCB, but each access method must be referenced for its meaning.

Offset	Bytes and Alignment	Field Name		Description
20(14)	1	DCBBUFNO		Number of buffers required for this data set. May range from 0 to a maximum of 255.
21(15)	.3	DCBBUFCB		Address of buffer pool control block.
24(18)	2	DCBBUFL		Length of buffer. May range from 0 to a maximum of 32760.
32(20)	1	DCBBFALN		Buffer Alignment:
		xx		Reserved bits.
		10	D	Doubleword boundary.
		01	F	Fullword not a doubleword boundary, coded in the DCB macro instruction.
32(20)	1	DCBBFTEK		Buffering technique:
		.xxx		Reserved bits.
		.1.0	S	Simple buffering.
		.110	A	QSAM locate mode processing of spanned records: OPEN is to construct a record area if it automatically constructs buffers.
		.010	R	BSAM create BDAM processing of unblocked spanned records: Software track overflow. OPEN forms a segment work area pool and stores the address of the segment work area control block in DCBECBW. However, WRITE uses a segment work area to write a record as one or more segments. BSAM input processing of unblocked spanned records with keys: Record offset processing. READ reads one record segment into the record area. The first segment of a record is preceded in the record area by the key. Subsequent segments are at an offset equal to the key length.
		.... x...		Reserved bit.
		.... 1...		XLRI being used to process a RECFM=DS or RECFM=DBS format tape data set (QSAM).
33(21)	.3	DCBEODAD		End-of-data address. Address of a user-provided routine to handle end-of-data conditions.

Offset	Bytes and Alignment	Field Name		Description
36(24)	1	DCBRECFM		Record format.
			Code	
		001.	D	Format-D record.
		10..	F	Fixed record length.
		01..	V	Variable record length.
		11..	U	Undefined record length.
		..1.	T	Track overflow.
		...1	B	Blocked records. May not occur with undefined (U).
		.... 1...	S	Fixed length record format: Standard blocks. (No truncated blocks or unfilled tracks are embedded in the data set.) Variable length record format: Spanned records.
		.... .10.	A	ISO/ANSI/FIPS control character.
		.... .01.	M	Machine control character.
		.... .00.		No control character.
		1		Key length (KEYLEN) was specified in the DCB macro instruction. This bit is inspected by the Open to prevent overriding a specification of KEYLEN=0 by a nonzero specification in the JFCB or data set label.
37(25)	.3	DCBEXLST		Exit list. Address of a user-provided exit list control block.
42(2A)	2	DCBMACRF		Macro instruction reference after OPEN.
				Contents and meaning are the same as those of the DCBMACR field in the foundation segment before OPEN.
50(32)	..2	DCBMACR (Before OPEN)		Major macro instructions and various options associated with them. Used by the Open routine to determine access method. Used by the access method executes in conjunction with other parameters to determine which load modules are required.
			Code	
		Byte 1		<u>BSAM—Input</u>
		00..		Always zero for BSAM.
		..1.	R	READ
		...x x..x		Reserved bits.
		.... .1..	P	POINT (which implies NOTE).
		.... ..1.	C	CNTRL
		Byte 2		<u>BSAM—Output</u>
51(33)		00..		Always zero for BSAM.
		..1.	W	WRITE
		.... 1...	L	Load mode BSAM (create BDAM data set).
		.... .1..	P	POINT (which implies NOTE).
		.... ..1.	C	CNTRL
		1		BSAM create BDAM processing of unblocked spanned records, with BFTEK=R specified: The user's program has provided a segment work area pool and stored the address of the segment work area control block in DCBEOBW.

Offset	Bytes and Alignment	Field Name		Description
		Byte 1		QSAM—Input
50(32)		0...		Always zero for QSAM.
		.1..	G	GET
		..0.		Always zero for QSAM.
		...1	M	Move mode.
		.... 1...	L	Locate mode.
		.... ..1.	C	CNTRL
		1	D	Data mode.
		Byte 2		QSAM—Output
51(33)		0...		Always zero for QSAM.
		.1..	P	PUT
		..0.		Always zero for QSAM.
		...1	M	Move mode.
		.... 1...	L	Locate mode.
		.... ..1.	C	CNTRL
		1	D	Data mode.
50(32)		Byte 1		BPAM—Input
		00..		Always zero for BPAM.
		..1.	R	READ
		.... .1..	P	POINT (which implies NOTE).
		...x x.xx		Reserved bits.
51(33)		Byte 2		BPAM—Output
		00..		Always zero for BPAM.
		..1.	W	WRITE
		.... .1..	P	POINT (which implies NOTE).
		...x x.xx		Reserved bits.

DIRECT ACCESS STORAGE DEVICE INTERFACE

Offset	Bytes and Alignment	Field Name	Description
16(10)	1	DCBKEYLE	Key length of the data set.
17(11)	.1	DCBDEVT	Device type.
		0010 0111	2305 Disk Storage Facility, Model 2.
		0010 1001	3330 Disk Storage, Model 1, or Mass Storage System (MSS) virtual volume.
		0010 1101	3330 Disk Storage, Model 11.
		0010 1010	3340/3344 Disk Storage.
		0010 1011	3350 Direct Access Storage.
		0010 1100	3375 Direct Access Storage.
		0010 1110	3380 Direct Access Storage.

MAGNETIC TAPE INTERFACE

Offset	Bytes and Alignment	Field Name	Description
16(10)	1	DCBTRTCH	Tape recording technique for 7-track tape.

	Code	
0010 0011	E	Even parity.
0011 1011	T	BCD/EBCDIC translation.
0001 0011	C	Data conversion.
0010 1011	ET	Even parity and translation.

Offset	Bytes and Alignment	Field Name	Description
17(11)	.1	DCBDEVT	Device type.

1000 0011	3400 series magnetic tape unit.
1000 1000	3480 Magnetic Tape Subsystem

Offset	Bytes and Alignment	Field Name	Description
18(12)	..1	DCBDEN	Tape density—3400 series magnetic tape units.

	Code	7-track	9-track	18-track
0100 0011	1	556 BPI	N/A	N/A
1000 0011	2	800 BPI	800 BPI	N/A
1100 0011	3	N/A	1600 BPI	N/A
1101 0011	4	N/A	6250 BPI	N/A

CARD READER, CARD PUNCH INTERFACE

Offset	Bytes and Alignment	Field Name	Description
16(10)	1	DCBMODE,DCBSTACK	

	Code	
1000	C	Column binary mode.
0100	E	EBCDIC mode.
.... xxxx		Stacker selection.
.... 0001	1	Stacker 1.
.... 0010	2	Stacker 2.
.... 0011	3	Stacker 3.

Offset	Bytes and Alignment	Field Name	Description
17(11)	.1	DCBDEVT	Device type.

0100 0001	2540 Card Reader
0100 0010	2540 Card Punch
0100 0100	2501 Card Reader
0100 0110	3505 Card Reader
0100 1100	3525 Card Punch

PRINTER INTERFACE

Offset	Bytes and Alignment	Field Name	Description
16(10)	1	DCBPRTSP	Number indicating normal printer spacing.
			Code
		0000 0000	0 No spacing.
		0000 0001	1 Space one line.
		0001 0001	2 Space two lines.
		0001 1001	3 Space three lines.
17(11)	.2	DCBDEVT	Device type.
		Byte 0	
		0100 1000	1403 Printer
		0100 1001	3211 Printer
		0100 1011	3203 Printer
		0100 1110	3800 Printing Subsystem
			Test-for-printer-overflow mask (PRTOV mask). If printer overflow is to be tested for, the PRTOV macro instruction sets the mask as follows:
		Byte 1	**Code**
		0010 0000	9 Test for channel 9 overflow.
		0001 0000	12 Test for channel 12 overflow.
19(13)	...1	DCBPRBYT	
		xxxx xx..	Reserved.
		11	Bits to identify presently active table reference character when 3800 printer is operating under OPTCD=J.

BSAM, BPAM Interface

Offset	Bytes and Alignment	Field Name	Description	
52(34)	1	DCBOPTCD	Option codes.	
			Code	
		1...	W	Write-validity check (DASD).
		.1..	U	Allow a data check caused by an invalid character. (1403 printer with UCS feature.) Window processing requested. (MSS)
			B	Treat EOF and EOV labels as EOV labels which allows SL or AL tapes to be read out of order. (Magnetic tape.)
		..1.	C	Chained scheduling. Input Tape Files: Requests the testing for and bypassing of any embedded DOS checkpoint records encountered. (This code can only be specified in a JCL statement.)
		.... 1...	Q	An ISCII/ASCII data set is to be processed.
		.... .1..	Z	Magnetic tape devices: Use reduced error recovery procedure.
		.... ..1.	T	BSAM only: user totaling.
		1	J	Specifies that the first data byte in the output data line will be a 3800 table reference character for dynamic selection of character sets.
57(39)	.3	DCBSYNAD	Address of user's synchronous error routine to be entered when a permanent error occurs.	
62(3E)	..2	DCBBLKSI	Maximum block size. Maximum value: 32760. The maximum block size for Version 3 ISO/ANSI/FIPS is 2048. An attempt to process from a Version 3 tape results in a label validation installation exit being taken. For fixed-length blocked record format, it must be a multiple of the length given in DCBLRECL. For variable-length records, this must include the 4-byte block length field. For more information about ISO/ANSI/FIPS spanned records, see Data Administration Guide.	
72(48)	1	DCBNCP	Number of channel programs. Number of READ or WRITE requests that may be issued prior to a CHECK. Maximum number: 99.	

Offset	Bytes and Alignment	Field Name	Description
80(50)	1	DCBUSASI/ DCBLBP	ISCII/ASCII tape. Block prefix.
		.1..	Block prefix is a 4-byte field containing the block length.
81(51)	.1	DCBBUFOF	Block prefix length.
82(52)	..2	DCBLRECL	Logical record length. For fixed-length blocked record format, the presence of DCBLRECL allows BSAM to read truncated records. For undefined records, this field contains block size.

Offset	Bytes and Alignment	Field Name	Code	Description
52(34)	1	DCBOPTCD		Option codes.
		1...	W	Write-validity check (DASD).
		.1..	U	Allow a data check for an invalid character. (1403 printer with UCS feature.) Window processing requested. (MSS)
			B	Treat EOF and EOV labels as EOV labels, which allows SL or AL tapes to be read out of order (magnetic tape).
		..1.	C	Chained scheduling.
		...1	H	Input Tape Files: Requests the testing for and bypassing of any embedded DOS checkpoint records encountered. (This code can only be specified in a JCL statement.)
		.... 1...	Q	An ISCII/ASCII data set is to be processed. Same as DCBOPTQ. BSAM only.
		.... .1..	Z	Magnetic tape devices. Use reduced error recovery procedure.
		.... ..1.	T	User totaling.
		1	J	Specifies that the first data byte in the output data line will be a 3800 table reference character.
57(39)	.3	DCBSYNAD		Address of the user's synchronous error routine to be entered when a permanent error occurs.
62(3E)	..2	DCBBLKSI		Maximum block size. Maximum value: 32760. The maximum block size for Version 3 ISO/ANSI/FIPS is 2048. An attempt to process from a Version 3 tape results in a label validation installation exit being taken. For fixed-length blocked record format, it must be a multiple of DCBRECL. For variable-length records this must include the 4-byte block length field provided by the access method.
80(50)	1	DCBUSASI/ DCBLBP		ISCII/ASCII tape. Block prefix.
		.1..		Block prefix is a 4-byte field containing the block length. (BUFOFF=L was specified).
81(51)	.1	DCBBUFOF		Block prefix length.

Offset	Bytes and Alignment	Field Name	Description
82(52)	..2	DCBLRECL	Format-F records: Record length. Format-U records: Block size. Format-V records — • Unspanned record format — GET: PUTX; record length. PUT: Actual or maximum record length. • Spanned record format — Locate mode — – GET: Segment length. – PUT: Actual or minimum segment length. Logical record interface — – Before OPEN: Maximum logical record length. – After GET: Record length. – Before PUT: Actual or maximum record length. – ISO/ANSI/FIPS spanned record format with XLRI; length of the record area in 'K' units (1024). Move mode — – GET: Record length. – PUT: Actual or maximum record length. • Data mode, GET — Data records up to 32752 bytes: Data length. Data records exceeding 32752 bytes: – Before OPEN: X'8000' – After OPEN: Data length. • Output mode, PUTX (output data set): Segment length.
84(54)	1	DCBEROPT	Error option. Disposition of permanent errors if the user returns from a synchronous error exit (DCBSYNAD), or if the user has no synchronous error exit.
		100.	ACC: Accept.
		010.	SKP: Skip.
		001.	ABE: Abnormal end of task.
		...x xxxx	Reserved bits.
85(55)	3	DCBCNTRA	Address of CNTRL module.
88(58)	2		Reserved.
90(5A)	2	DCBPRECL	Block length, maximum block length, or data length.
92(5C)	4	DCBEOB	Address of end of block module.

DATA CONTROL BLOCK—ISAM

Offset	Bytes and Alignment	Field Name	Description
16(10)	1	DCBKEYLE	Key length.
17(11)	.1	DCBDEVT	Device type.
		0000 0111	2305 Disk Storage Facility, Model 2.
		0000 1001	3330 Disk Storage, Model 1, or Mass Storage System (MSS) virtual volume.
		0000 1101	3330 Disk Storage, Model 11.
		0000 1010	3340 Disk Storage.
		0000 1011	3350 Direct Access Storage.
		0010 1100	3375 Direct Access Storage.
		0010 1110	3380 Direct Access Storage, all models.
20(14)	1	DCBBUFNO	Number of buffers required for this data set: 0-255.
21(15)	.3	DCBBUFCB	Address of buffer pool control block.
24(18)	2	DCBBUFL	Length of buffer: 0 - 32760 bytes.
32(20)	1	DCBBFALN	Buffer alignment:
			Code
		xx	Reserved bits.
		10	D Doubleword boundary.
		01	F Fullword not a doubleword boundary, coded in the DCB macro instruction.
		11	F Fullword not a doubleword boundary, coded in the DD statement.
33(21)	.3	DCBEODAD	Address of a user-provided routine to handle end-of-data conditions.
36(24)	1	DCBRECFM	Record format.
			Code
		10..	F Fixed length records.
		10..	V Variable length records.
		11..	U Undefined length records.
		..1.	T Track overflow.
		...1	B Blocked records. May not occur with undefined (U).
		.... 1...	S Standard records. No truncated blocks or unfilled tracks are embedded in the data set.
		.... .10.	A ISO/ANSI/FIPS control character.
		.... .01.	M Machine control character.
		.... .00.	No control character.
		1	Key length (KEYLEN) was specified in the DCB macro instruction; this bit is inspected by the open routine to prevent overriding a specification of KEYLEN=0 by a nonzero specification in the JFCB or data set label.
37(25)	.3	DCBEXLST	Exit list. Address of a user-provided list.
42(2A)	..2	DCBMACRF	Macro instruction reference after OPEN:
			Contents and meaning are the same as those of the DCBMACR field before OPEN.

Offset	Bytes and Alignment	Field Name		Description
50(32)	..2	DCBMACR		Macro instruction reference before OPEN: specifies the major macro instructions and various options associated with them. Used by the Open routine to determine access method. Used by the access method executors in conjunction with other parameters to determine which load modules are required.
			Code	
		Byte 1		<u>BISAM</u>
50(32)		00.0 0...		Always zero for BISAM.
		..1.	R	READ
		.... .1..	S	Dynamic buffering.
		.... ..1.	C	CHECK
		x		Reserved bit.
		Byte 2		<u>BISAM</u>
51(33)		00.0 0000		Always zero for BISAM.
		..1.	W	WRITE
		Byte 1		<u>QISAM</u>
50(32)		0.0. .0..		Always zero for QISAM.
		.1..	G	GET
		...1	M	Move mode of GET.
		.... 1...	L	Locate mode for GET.
		xx		Reserved bits.
		Byte 2		<u>QISAM</u>
51(33)		1...	S	SETL
		.1..	P	PUT or PUTX.
		..0.		Always zero for QISAM.
		...1	M	Move mode of PUT.
		.... 1...	L	Locate mode of PUT.
		.... .1..	U	Update in place (PUTX).
		.... ..1.	K	SETL by key.
		1	I	SETL by ID.
52(34)	1	DCBOPTCD		Option codes:
			Code	
		1...	W	Write-validity check.
		.1..	U	Full-track index write.
		..1.	M	Master indexes.
		...1	I	Independent overflow area.
		.... 1...	Y	Cylinder overflow area.
		.... ..1.	L	Delete option.
		1	R	Reorganization criteria.
		.... .x..		Reserved bit.
53(35)	.1	DCBMAC		Extension of the DCBMACRF field for ISAM.
			Code	
		xxxx ...x		Reserved bits.
		.... 1...	U	Update for read.
		.... .1..	U	Update type of write.
		.... ..1.	A	Add type of write.
54(36)	..1	DCBNTM		Number of tracks that determines the development of a master index. Maximum permissible value: 99.
55(37)	...1	DCBCYLOF		The number of tracks to be reserved on each prime data cylinder for records that overflow from other tracks on that cylinder. To determine how to calculate the maximum number, see the section on allocating space for an ISAM data set in <u>Data Administration Guide</u>.

Offset	Bytes and Alignment	Field Name	Description
56(38)	4	DCBSYNAD	Address of user's synchronous error routine to be entered when uncorrectable errors are detected in processing data records.
60(3C)	2	DCBRKP	Relative position of the first byte of the key within each logical record. Maximum permissible value: logical record length minus key length.
62(3E)	..2	DCBBLKSI	Block size.
64(40)	4	DCBMSWA	Address of the storage work area reserved for use by the control program when new records are being added to an existing data set.
68(44)	2	DCBSMSI	Number of bytes in area reserved to hold the highest level index.
70(46)	2	DCBSMSW	Number of bytes in work area used by control program when new records are being added to the data set.
72(48)	1	DCBNCP	Number of copies of the READ-WRITE (type K) channel programs that are to be established for this data control block (99 maximum).
73(49)	.3	DCBMSHI	Address of the storage area holding the highest level index.
80(50)	1	DCBEXCD1	First byte in which exceptional conditions detected in processing data records are reported to the user.
		1...	Lower key limit not found.
		.1..	Invalid device address for lower limit (QISAM only). Record length check (BISAM only).
		..1.	Space not found.
		...1	Invalid request.
		.... 1...	Uncorrectable input error.
		.... .1..	Uncorrectable output error (BISAM only). Block could not be reached (BISAM only).
		.... ..1.	Block could not be reached (input) (QISAM only). Overflow record (BISAM only).
		1	Block could not be reached (update) (QISAM only) Duplicate record (BISAM only)
81(51)	.1	DCBEXCD2	Second byte in which exceptional conditions detected in processing data records are reported to the user (QISAM only).
		1...	Sequence check.
		.1..	Duplicate record.
		..1.	DCB closed when error was detected.
		...1	Overflow record.
		.... 1...	PUT: length field of record larger than length indicated in DCBLRECL.
		xxx	Reserved bits.
82(52)	..2	DCBLRECL	Logical record length for fixed-length record formats. Variable-length record formats: maximum logical record length or an actual logical record length changed dynamically by the user when creating the data set.

Offset	Bytes and Alignment	Field Name	Description
197(C5)	.1	DCBOVDEV	Device type for independent overflow.
		0000 0111	2305 Disk Storage Facility, Model 2.
		0000 1001	3330 Disk Storage, Model 1, or Mass Storage System (MSS) virtual volume.
		0000 1101	3330 Disk Storage, Model 11.
		0000 1010	3340/3344 Disk Storage.
		0000 1011	3350 Direct Access Storage.
		0010 1100	3375 Direct Access Storage.
		0010 1110	3380 Direct Access Storage.

Offset	Bytes and Alignment	Field Name		Description
16(10)	1	DCBKEYLE		Key length.
17(11)	.3	DCBREL		Number of relative tracks or blocks in this data set.
20(14)	1	DCBBUFNO		Number of buffers required for this data set. May range from 0 to 255.
21(15)	.3	DCBBUFCB		Address of buffer pool control block or of dynamic buffer pool control block.
24(18)	2	DCBBUFL		Length of buffer. May range from 0 to 32760.
32(20)	1	DCBBFALN		Buffer alignment:
		xx		Reserved bits
		10		Doubleword boundary.
		01		Fullword not a doubleword boundary, coded in the DCB macro instruction.
		11		Fullword not a doubleword boundary, coded in the DD statement.
		.x.x x...		Reserved bits.
32(20)	1	DCBBFTEK		Buffering technique.
		..x.		Reserved bit.
		..1.	R	Unblocked spanned records: Variable spanned record format. Open forms a segment work area pool. The number of segment work areas is determined by DCBBUFNO (OPEN stores the address of the segment work area control block in DCBDYNB if dynamic buffering is not used or in the dynamic buffer pool control block (see DCBBUFCB) if dynamic buffering is used. WRITE uses a segment work area to write a record as one or more segments. READ uses a segment work area to read a record that was written as one or more segments.
36(24)	1	DCBRECFM		Record format.
			Code	
		10..	F	Fixed record length.
		01..	V	Variable record length.
		11..	U	Undefined record length.
		..1.	T	Track overflow.
		...1	B	Blocked (allowed only with V).
		.... 1...	S	Spanned (allowed only with V).
		.... .00.		Always zeros.
		1		Key length (KEYLEN) was specified in the DCB macro instruction. This bit is inspected by the open routine to prevent overriding a specification of KEYLEN=0 by a nonzero specification in the JFCB or data set label.
37(25)	.3	DCBEXLST		Exit list. Address of a user-provided exit list control block.

Offset	Bytes and Alignment	Field Name		Description
42(2A)	..2	DCBMACRF		Macro instruction reference after OPEN.
				Contents and meaning are the same as DCBMACR before OPEN.
50(32)	..2	DCBMACR		Macro instruction reference before OPEN: major macro instructions and various options associated with them that will be used.
		Byte 1	**Code**	
50(32)		00..		Always zero for BDAM.
		..1.	R	READ
		...1	K	Key segment with READ.
		.... 1...	I	ID argument with READ.
		.... .1..	S	System provides area for READ (dynamic buffering).
		.... ..1.	X	Read exclusive.
		1	C	CHECK macro instruction.
		Byte 2	**Code**	
51(33)		00..		Always zero for BDAM.
		..1.	W	WRITE
		...1	K	Key segment with WRITE.
		.... 1...	I	ID argument with WRITE.
		.... .x..		Reserved bit.
		.... ..1.	A	Add type of WRITE.
		1		Unblocked spanned records, with BFTEK=R specified and no dynamic buffering: The user's program has provided a segment work area pool and stored the address of the segment work area control block in DCBDYNB.
52(34)	1	DCBOPTCD		Option codes:
			Code	
		1...	W	Write-validity check.
		.1..		Track overflow.
		..1.	E	Extended search.
		...1	F	Feedback.
		.... 1...	A	Actual addressing.
		.... .1..		Dynamic buffering.
		.... ..1.		Read exclusive.
		1	R	Relative block addressing.
56(38)	4	DCBSYNAD		Address of SYNAD (synchronous error) routine.
62(3E)	..2	DCBBLKSI		Maximum block size.
81(51)	.3	DCBLIMCT		Number of tracks or number of relative blocks to be searched (extended search option).

APPENDIX G. PDABD SYMBOLIC FIELD NAMES

The following describes PDABD fields of the dummy control section generated by the PDABD macro instruction. Included are the names, attributes, and descriptions of the dummy control section. The use of any of the symbolic names provided by the dummy section should be preceded by a USING instruction specifying IHAPDAB and a dummy section base register containing the address of the actual parallel data access block.

```
               PDABD
IHAPDAB        DSECT
PDANODCB       DS         H    Number of DCB addresses in list
PDAMAXCB       DS         H    Maximum number of addresses allowed
PDAGRTNA       DS         A    Address of parallel GET routine
PDADCBAI       DS         F    DCB address increment
PDADCBLA       DS         A    Address of last DCB entry
PDADCBEP       DS         A    Address of DCB entry last processed
PDAECBIX       DS         F    Index to ECB list
PDADCBAL       EQU        *    Start of DCB list
```

GLOSSARY OF TERMS AND ABBREVIATIONS

The following terms are defined as they are used in this book. If you do not find the term you are looking for, refer to the index or to the IBM Vocabulary for Data Processing, Telecommunications, and Office Systems, GC20-1699.

ABE. abnormal end (value of EROPT)

ABEND. abnormal end (macro instruction)

ABSTR. absolute track (value of SPACE)

ACC. accept erroneous block (value of EROPT)

access method. A technique for moving data between main storage and input/output devices.

address marker. A byte of data on a disk or diskette, used to identify the data field and ID field in the record.

AFF. affinity (channel separation parameter of DD statement or unit affinity value of UNIT)

AL. American National Standard Labels (Value of LABEL).

alias. (1) An alternate label. For example, a label and one or more aliases may be used to refer to the same data element or point in a computer program. (2) An alternate name for a member of a partitioned data set.

ANSI. American National Standards Institute

ASCII. American National Standard Code for Information Interchange

AUL. American National Standard user labels (value of LABEL)

auxiliary storage. Data storage other than virtual storage; for example, storage on magnetic tape or direct access devices.

basic access technique. Any access method in which input/output statement causes a corresponding machine input/output operation to occur. Contrast with queued access technique.

BCD. binary coded decimal

BCDIC. binary coded decimal interchange code

BDAM. basic direct access method

BDW. block descriptor word

BFALN. buffer alignment (operand of DCB)

BFTEK. buffer technique (operand of DCB)

BISAM. basic indexed sequential access method

BLDL. build list (macro instruction)

BLKSIZE. block size (operand of DCB)

block prefix. An optional variable length field that may precede unblocked records or blocks of records in ASCII on magnetic tapes.

block size. (1) The number of records, words, or characters in a block. (2) A measure of the size of a block, usually specified in units such as records, words, computer words, or characters.

blocking. The process of combining two or more records into one block.

BPAM. basic partitioned access method

BPI. bits per inch

BSAM. basic sequential access method

BSM. backspace past tapemark and forward space over tapemark (operand of CNTRL)

BSP. backspace one block (macro instruction)

BSR. backspace over a specified number of blocks (operand of CNTRL).

BUFCB. buffer pool control block (operand of DCB)

buffer. An area of storage that is temporarily reserved for use in performing an input/output operation, into which data is read or from which data is written.

buffer pool. An area of storage in which all buffers of a program are kept; the buffers are built in extents chained together.

BUFL. buffer length (operand of DCB)

BUFNO. buffer number (operand of DCB)

BUFOFF. buffer offset (length of ASCII block prefix by which the buffer is offset; operand of DCB)

CCHH. The cylinder head record that gives the DASD location.

CCW. channel command word

channel program. One or more channel command words that control a specific sequence of data channel operations. Execution of the specific sequence is initiated by a single start I/O instruction.

CNTRL. control (macro instruction)

CONTIG. contiguous space allocation (value of SPACE)

control character. A character whose occurrence in a particular context initiates, modifies, or stops a control operation. It may be recorded for use in a subsequent action, and may have a graphic representation in some circumstances.

control program. A routine, usually part of an operating system, that aids in controlling the operations and managing the resources of a computer system.

CSW. channel status word

CVOL. See OS CVOL.

cylinder. (1) In a disk pack, the set of all tracks with the same nominal distance from the axis about which the disk pack rotates. (2) The tracks of a disk storage device that can be accessed without repositioning the access mechanism.

CYLOFL. number of tracks for cylinder overflow records (operand of DCB)

D. format-D (ISCII/ASCII variable-length) records (value of RECFM)

DA. direct access (value of DEVD or DSORG)

DASD. See direct access storage device

data control block. A control block used by access method routines in storing and retrieving data.

data conversion. The process of changing data from one form of representation to another.

data definition (DD) statement. A job control statement that describes a data set associated with a particular job step.

data extent block. An extension of the data control block that contains information about the physical status of the data set being processed.

data set. The major unit of data storage and retrieval in the operating system, consisting of data in a prescribed arrangement and described by control information to which the system has access. As used in this publication, a collection of fixed- or variable-length records in auxiliary storage.

data set control block. A data set label for a data set in direct access storage.

data set label. A collection of information that describes the attributes of a data set and is normally stored on the same volume as the data set.

DAU. direct access unmovable data set (value of DSORG)

DB. ISCII/ASCII variable-length, blocked records (value of RECFM)

DBS. ISCII/ASCII variable-length, blocked spanned records (value of RECFM)

DCB. See data control block (control block name, macro instruction, or parameter on DD statement)

DCBD. data control block dummy section (macro instruction)

DD. data definition

DEB. See data extent block

DECB. data event control block

DEN. magnetic tape density (operand of DCB)

dequeue. To remove items from a queue. Contrast with enqueue.

DEVD. Device-dependent (operand of DCB)

direct access storage device. A device in which the access time is effectively independent of the location of the data.

direct data set. A data set whose records are in random order on a direct access volume. Each record is stored or retrieved according to its actual address or its address according to the beginning of the data set. Contrast with sequential data set.

directory. (1) A table of identifiers and references to the corresponding items of data. (2) An index that is used by a control program to locate one or more blocks of data that are stored in separate areas of data set in direct access storage.

DISP. data set disposition (parameter of DD statement)

doubleword. A contiguous sequence of bits or characters that comprises two computer words and is capable of being addressed as a unit.

DS. ISCII/ASCII variable-length, spanned records (value of RECFM)

DSCB. See data set control block

DSECT. See dummy control section.

DSORG. data set organization (operand of DCB)

dummy control section. A control section that an assembler can use to format an area of storage without producing any object code.

dynamic allocation. The allocation of a data set or volume by the use of the data set name or volume serial number rather than by the use of information contained in a JCL statement.

dynamic buffering. (1) A dynamic allocation of buffer storage. (2) Allocation of storage for buffers as they are needed for incoming data during program execution.

EBCDIC. extended binary coded decimal interchange code

enqueue. To place items on a queue. Contrast with dequeue.

entry point. (1) The address or the level of the first instruction executed upon entering a computer program, a routine, or a subroutine. A computer program, a routine, or a subroutine may have a number of different entry points, each perhaps corresponding to a different function or purpose. (2) In a routine, any place to which control can be passed.

EODAD. End-of-data set exit routine address (operand of DCB)

EOF. end-of-file

EOV. end-of-volume

EROPT. error options (operand of DCB)

ESETL. end sequential retrieval (QISAM macro instruction)

EXCP. execute channel program (macro instruction)

exit list. A control block that contains the addresses of routines that receive control when specified events occur during execution; for example, routines that handle session establishment request processing or I/O errors.

EXLST. See exit list (operand of DCB)

F. fixed-length records (value of RECFM)

FB. fixed-length, blocked records (value of RECFM)

FBS. fixed-length, blocked, standard records (value of RECFM)

FBT. fixed-length, blocked records with track overflow option (value of RECFM)

FCB. forms control buffer

FEOV. force end-of-volume (macro instruction)

FIPS. Federal Information Processing Standard

flag. (1) Any of various types of indicators used for identification, for example, a wordmark. (2) A character that signals the occurrence of some condition, such as the end of a word.

format-D. ISCII/ASCII or ISO/ANSI/FIPS variable-length records

format-F. Fixed-length records

format-U. Undefined-length records

format-V. Variable-length records

FS. fixed-length, standard records (value of RECFM)

FSM. forward space past tapemark and backspace over tapemark (operand of CNTRL)

FSR. forward space over a specified number of blocks (records) (operand of CNTRL)

GCR. group coded recording (tape recording mode)

generation data group. A collection of data sets that are kept in chronological order; each data set is called a generation data set.

generation data set. One of a collection of historically related non-VSAM data sets; the collection of these data sets is known as a generation data group.

GL. GET macro, locate mode (value of MACRF)

GM. GET macro, move mode (value of MACRF)

HA. home address

halfword. A contiguous sequence of bits or characters that comprise half a computer word and is capable of being addressed as a unit.

head. A device that reads, writes, or erases data on a storage medium, for example, a small electromagnet used to read, write, or erase data on magnetic drum or magnetic tape, or the set of perforating, reading, or marking devices used for punching, reading, or printing on perforated tape.

header label. (1) An internal label, immediately preceding the first record of a file, that identifies the file and contains data used in file control. (2) The label or data set label that precedes the data records on a unit of recording media.

home address. An address written on a direct access volume, denoting a track's address relative to the beginning of the volume.

IBG. Inter-block gap

ICF catalog. integrated catalog facility catalog

INOUT. input then output (operand of OPEN)

integrated catalog facility (ICF). The name of the catalog associated with the Data Facility Product licensed program.

internal storage. Storage that is accessible by a computer without the use of input/output channels.

I/O. input/output

IOB. input/output block

IPL. initial program load

IRG. interrecord gap

IS. indexed sequential (value of DSORG)

ISAM. indexed sequential access method

ISCII. International Standard Code for Information Interchange

ISO. International Organization for Standardization

ISU. indexed sequential unmovable (value of DSORG)

JCL. job control language

JFCB. job file control block

JFCBE. job file control block extension

K. 1024 (bytes)

key. One or more consecutive characters taken from a data record, used to identify the record and establish its order with respect to other records.

KEYLEN. key length (operand of DCB)

locate mode. A way of providing data by pointing to its location instead of moving it.

logical record. (1) A record from the standpoint of its content, function, and use rather than its physical attributes; that is, one that is defined in terms of the information it contains. (2) A unit of information normally pertaining to a single subject; a logical record is that user record requested of or given to the data management function.

LPA. link pack area

LPALIB. link pack area library

LRECL. logical record length (operand of DCB)

LRI. logical record interface

M. machine control code (value of RECFM)

MACRF. macro instruction form (operand of DCB)

master catalog. A key-sequenced data set with an index containing extensive data set and volume information required to locate data sets, to allocate and deallocate storage space, to verify the authorization of a program or operator to gain access to a data set, and to accumulate usage statistics for data sets.

member. A partition of a partitioned data set.

MOD. modify data set (value of DISP)

move mode. A transmittal mode in which the record to be processed is moved into a user work area.

MSHI. main storage for highest-level index (operand of DCB)

MSS. IBM 3850 Mass Storage System

MSVC. Mass Storage Volume Control

MSWA. main storage for work area (operand of DCB)

NCP. number of channel programs (operand of DCB)

non-VSAM data set. A data set created and accessed using one of the following methods: BDAM, BPAM, BSAM, QSAM, QISAM.

NOPWREAD. No password required to read a data set (value of LABEL)

NRZI. non-return-to-zero-inverted (tape recording mode)

NSL. nonstandard label (value of LABEL)

NTM. number of tracks in cylinder index for each entry in lowest level of master index (operand of DCB)

operand. Information entered with a command name to define the data on which a command operates and to control the execution of the command.

operating system. Software that controls the execution of programs; an operating system may provide services such as resource allocation, scheduling, input/output control, and data management.

OPTCD. optional services code (operand of DCB)

OS CVOL. operating system control volume

OUTIN. output then input (operand of OPEN)

partitioned data set. A data in direct access storage that is divided into partitions, called members, each of which can contain a program, part of a program, or data. Synonymous with program library.

password. A unique string of characters stored in a catalog that a program or a computer operator at the console must supply to meet security requirements before the program gains access to a data set.

PCI. program-controlled interruption

PDAB. parallel data access block

PDS. partitioned data set

PE. phase encoding (tape recording mode)

physical record. One or more logical records, or occasionally, a part of one logical record read into or written from main storage as a unit.

PL. PUT macro, locate mode (value of MACRF)

PM. PUT macro, move mode (value of MACRF)

PO. partitioned organization (value of DSORG)

pointer. An address or other indication of location.

POU. partitioned organization unmovable (value of DSORG)

problem program. Any program that is executed when the processing unit is in the problem state; that is, any program that does not contain privileged instructions. This includes IBM-distributed programs, such as language translators and service programs, as well as programs written by a user.

PRTSP. printer line spacing (operand of DCB)

PS. physical sequential (value of DSORG)

PSU. physical sequential unmovable (value of DSORG)

QISAM. queued indexed sequential access methods

QSAM. queued sequential access method

queued access technique. Any access method that synchronizes the transfer of data between the computer program using the access method and input/output devices, thereby minimizing delays for input/output operations.

RACF. See Resource Access Control Facility.

RDBACK. read backward (operand of OPEN)

RDW. record descriptor word

RECFM. record format (operand of DCB)

record. A collection of related data or words, treated as a unit; for example, in stock control, each invoice could constitute one record.

register. An internal computer component capable of storing a specified amount of data and accepting or transferring this data rapidly.

relative address. An address expressed as a difference with respect to a base address.

Resource Access Control Facility. A licensed program that provides for access control by identifying and verifying users to the system authorizing access to DASD data sets, logging detected unauthorized attempts to enter the system, and logging detected accesses to protected data sets.

RKP. relative key position (operand of DCB)

RLSE. release unused space (DD statement)

RPS. rotational position sensing

save area. An area of main storage in which the contents of registers are saved.

scheduling. The ability to request that a task set should be started at a particular time interval or on occurrence of a specified PI interrupt.

SDW. segment descriptor word

secondary space. An area of direct access storage space which is allocated after the primary space originally allocated has been exhausted.

sequential data set. A data set whose records are organized on the basis of their successive physical positions, such as on magnetic tape. Contrast with direct data set.

SER. volume serial number (value of VOLUME)

serialization. In MVS, the prevention of a program from using a resource that is already being used by an interrupted program until the interrupted program is finished using the resource.

SETL. set lower limit of sequential retrieval (QISAM macro instruction)

SF. sequential forward (operand of READ or WRITE)

SK. skip to a printer channel (operand of CNTRL)

SKP. skip erroneous block (value of EROPT)

SL. IBM standard labels (value of LABEL)

SMSI. size of main-storage area for highest-level index (operand of DCB)

SMSW. size of main-storage work area (operand of DCB)

SP. space lines on a printer (operand of CNTRL)

spooling. (1) The use of auxiliary storage as a buffer to reduce processing delays when transferring data between peripheral equipment and the processors of a computer. (2) The reading of input data streams and the output of data streams on auxiliary storage devices, concurrently with job execution, in a format convenient for later processing or output operations.

SS. select stacker on card reader (operand of CNTRL)

substitute mode. A transmittal mode used with exchange buffering on which segments are pointed to, and exchanged with, user work areas.

subtask. (1) A task that is initiated and terminated by a higher order task. (2) A task that is restricted from communication with an operator device.

SUL. IBM standard and user labels (value of LABEL)

SVC. supervisor call

SVCLIB. supervisor call library

SYNAD. synchronous error routine address (operand of DCB)

SYSIN. system input stream

SYSOUT. system output stream

system residence volume. The volume on which the nucleus of the operating system and the highest level index of the catalog are located.

T. track overflow option (value of RECFM); user-totaling (value of OPTCD)

TIOT. task I/O table

trailer label. A file or data set label that follows the data records on a unit of recording media.

TRC. table reference character

TRTCH. track recording technique (operand of DCB)

TSO. Acronym of time sharing option.

TTR. Acronym of track record.

U. undefined length records (value of RECFM)

UCS. universal character set

UHL. user header label

user catalog. A catalog used in the same way as the master catalog, but optional and pointed to by the master catalog, and also used to lessen the contention for the master catalog and to facilitate volume portability.

UTL. user trailer label

V. format-V (variable-length) records (value of RECFM)

VB. variable-length, blocked records (value of RECFM)

VBS. variable-length, blocked, spanned records (value of RECFM)

virtual I/O (VIO). A facility that pages data into and out of external page storage; to the problem program, the data to be read from or written to direct access storage devices.

volume. (1) A data carrier that is mounted and demounted as a unit, for example, a reel of magnetic tape, a disk pack. (2) That portion of a single unit of storage that is accessible to a single read/write mechanism, for example, a drum, a disk pack, or a part of a disk module.

volume table of contents. A table on a direct access volume that describes each data set on the volume.

VS. variable-length, spanned records

VSAM catalog. virtual storage access method catalog

VTOC. volume table of contents

XLRI. extended logical record interface

INDEX

Reader's Comment Form

MVS/XA Data Administration:
Macro Instruction Reference
GC26-4141-2

This manual is part of a library that serves as a reference source for system analysts, programmers, and operators of IBM systems. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Your comments will be sent to the author's department for whatever review and action, if any, are deemed appropriate.

Note: Do not use this form to request IBM publications. If you do, your order will be delayed because publications are not stocked at the address printed on the reverse side. Instead, you should direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.

If you wish a reply, give your name, company, mailing address, and telephone number.

____

____

____

____

If you have applied any technical newsletters (TNLs) to this book, please list them here:

Last TNL __________________

Previous TNL __________________

Fold on two lines, tape, and mail. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.)
Thank you for your cooperation.

Note: Staples can cause problen h automated mail sorting equipment.
Please use pressure sensitive or other gummed tape to seal this form.

GC26-4141-2

Reader's Comment Form

Fold and tape

Please do not staple

Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
P.O. Box 50020
Programming Publishing
San Jose, California 95150

Fold and tape

Please do not staple

Fold and tape

MVS/XA Data Administration: Macro Instruction Reference (File No. S370-34) Printed in U.S.A. GC26-4141-2

IBM®

MVS/Extended Architecture
Data Administration:
Macro Instruction Reference

File Number S370-34

GC26-4141-02

Printed in U.S.A.