SC26-4506-1

MVS/ESA
Data Administration:
Macro Instruction Reference

Version 3 Release 1

MVS/ESA
Data Administration:
Macro Instruction Reference

SC26-4506-1

Version 3 Release 1

Second Edition (June 1989)

This edition replaces and makes obsolete the previous edition, SC26-4506-0.

This edition applies to Version 3 Release 1 of MVS/DFP™, Program Number 5665-XA3, 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 table of contents. Specific changes are indicated by a vertical bar to the left of the change. A vertical bar to the left of a figure caption indicates that the figure has changed. 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, 4300, and 9370 Processors Bibliography*, GC20-0001, for the editions that are applicable and current.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM licensed program in this publication is not intended to state or imply that only IBM's program may be used. Any functionally equivalent program may be used instead.

Requests for IBM publications should be made to your IBM representative or to the IBM branch office serving your locality. If you request publications from the address given below, your order will be delayed because publications are not stocked there.

A Reader's Comment Form is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Department J57, P. O. Box 49023, San Jose, California, U.S.A. 95161-9023. IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Trademarks

The following names have been adopted by IBM for trademark use and are used throughout this publication:

ESA/370™

MVS/DFP™

MVS/ESA™

MVS/SP™

Contents

Introduction 1
24- and 31-Bit Addressing Considerations 1
ISO, ANSI, and FIPS Labels 1
Notational Conventions 2
Macro Instruction Format 4
Rules for Register Usage 6
Rules for Continuation Lines 7

Macro Instruction Descriptions 9
BLDL—Build a Directory Entry List (BPAM) 9
Completion Codes 11
BSP—Backspace a Physical Record (BSAM—Magnetic Tape and Direct Access Only) 12
Completion Codes 13
BUILD—Build a Buffer Pool (BDAM, BISAM, BPAM, BSAM, QISAM, and QSAM) 14
BUILDRCD—Build a Buffer Pool and a Record Area (QSAM) 16
BUILDRCD—List Form 18
BUILDRCD—Execute Form 19
CHECK—Wait for and Test Completion of a Read or Write Operation (BDAM, BISAM, BPAM, and BSAM) 20
CHKPT—Take a Checkpoint for Restart within a Job Step 22
CLOSE—Logically Disconnect a Data Set (BDAM, BISAM, BPAM, BSAM, QISAM, and QSAM) 23
CLOSE—List Form 26
CLOSE—Execute Form 28
Return Codes from CLOSE 29
CNTRL—Control Online Input/Output Device (BSAM and QSAM) 30
DCB—Construct a Data Control Block (BDAM) 33
DCB—Construct a Data Control Block (BISAM) 42
DCB—Construct a Data Control Block (BPAM) 48
DCB—Construct a Data Control Block (BSAM) 55
DCB—Construct a Data Control Block (QISAM) 72
DCB—Construct a Data Control Block (QSAM) 80
DCBD—Provide Symbolic Reference to Data Control Blocks (BDAM, BISAM, BPAM, BSAM, QISAM, and QSAM) 98
ESETL—End Sequential Retrieval (QISAM) 100
FEOV—Force End-of-Volume (BSAM and QSAM) 101
FIND—Establish the Beginning of a Data Set Member (BPAM) 102
Completion Codes 102
FREEBUF—Return a Buffer to a Pool (BDAM, BISAM, BPAM, and BSAM) . 104
FREEDBUF—Return a Dynamically Obtained Buffer (BDAM and BISAM) . 105
FREEPOOL—Release a Buffer Pool (BDAM, BISAM, BPAM, BSAM, QISAM, and QSAM) 106
GET—Obtain Next Logical Record (QISAM) 107
GET—Obtain Next Logical Record (QSAM) 108
GET Routine Exits 110
GETBUF—Obtain a Buffer (BDAM, BISAM, BPAM, and BSAM) 111
GETPOOL—Build a Buffer Pool (BDAM, BISAM, BPAM, BSAM, QISAM, and QSAM) 112
MSGDISP—Displaying a Ready Message 113

MSGDISP—List Form 114
MSGDISP—Execute Form 115
Completion Codes 116
NOTE—Provide Relative Position (BPAM and BSAM—Tape and Direct Access Only) 117
Completion Codes—If Type=ABS is Specified 118
Completion Codes—If Type=REL is Specified 118
OPEN—Logically Connect a Data Set (BDAM, BISAM, BPAM, BSAM, QISAM, and QSAM) 119
Return Codes from OPEN 123
OPEN—List Form 125
OPEN—Execute Form 127
PDAB—Construct a Parallel Data Access Block (QSAM) 129
PDABD—Provide Symbolic Reference to a Parallel Data Access Block (QSAM) 130
POINT—Position to a Relative Block (BPAM and BSAM—Tape and Direct Access Only) 131
Completion Codes 133
PRTOV—Test for Printer Carriage Overflow (BSAM and QSAM—Online Printer and 3525 Card Punch, Print Feature) 134
PUT—Write Next Logical Record (QISAM) 136
PUT Routine Exit 137
PUT—Write Next Logical Record (QSAM) 138
PUT Routine Exit 140
PUTX—Write a Record from an Existing Data Set (QISAM and QSAM) . . . 141
PUTX Routine Exit 141
READ—Read a Block (BDAM) 142
READ—Read a Block of Records (BISAM) 145
READ—Read a Block (BPAM and BSAM) 147
READ—Read a Block (Offset Read of Keyed BDAM Data Set Using BSAM) 149
READ—List Form 150
READ—Execute Form 151
RELEX—Release Exclusive Control (BDAM) 152
Completion Codes 152
RELSE—Release an Input Buffer (QISAM and QSAM Input) 153
SETL—Set Lower Limit of Sequential Retrieval (QISAM Input) 154
SETL Exit 155
SETPRT—Printer Setup (BSAM, QSAM, and EXCP) 156
3800 Printers and SYSOUT Data Sets 156
Non-3800 Printers 156
4248 Printers 157
All Printers 157
Return Codes 164
Reason Codes 170
SETPRT—List Form 173
SETPRT—Execute Form 175
STOW—Update Partitioned Data Set Directory (BPAM) 178
Completion Codes 180
SYNADAF—Perform SYNAD Analysis Function (BDAM, BISAM, BPAM, BSAM, EXCP, QISAM, and QSAM) 182
Completion Codes 184
Message Buffer Format 185
SYNADRLS—Release SYNADAF Buffer and Save Areas (BDAM, BISAM, BPAM, BSAM, EXCP, QISAM, and QSAM) 187
SYNCDEV—Synchronize Device 188

Tape Data Sets . . . 188
SYNCDEV—List Form . . . 190
SYNCDEV—Execute Form . . . 191
Completion Codes . . . 192
TRUNC—Truncate an Output Buffer (QSAM Output—Fixed- or Variable-Length Blocked Records) . . . 193
WAIT—Wait for One or More Events (BDAM, BISAM, BPAM, and BSAM) . 194
WRITE—Write a Block (BDAM) . . . 196
WRITE—Write a Logical Record or Block of Records (BISAM) . . . 199
WRITE—Write a Block (BPAM and BSAM) . . . 201
WRITE—Write a Block (Create a Direct Data Set with BSAM) . . . 203
Completion Codes for WRITE—Write a Block (Create a Direct Data Set with BSAM) . . . 205
WRITE—List Form . . . 206
WRITE—Execute Form . . . 207
XLATE—Translate to and from ISCII/ASCII (BSAM and QSAM) . . . 208

Appendix A. Status Information Following an Input/Output Operation . . . 209
Data Event Control Block . . . 209

Appendix B. Data Management Macro Instructions Available by Access Method . . . 211

Appendix C. Device Capacities . . . 213
Card Readers and Card Punches . . . 213
Printers . . . 213
Magnetic Tape Units . . . 213
Direct Access Devices . . . 214
Track Capacity Determination . . . 214
Space Calculation Formulas for Models 2305 through 3375 . . . 214
3380 (All Models) . . . 215
Track Capacity Calculation Examples for 3380 Models . . . 217
Equal Length Records . . . 217
Unequal Length Records . . . 218

Appendix D. DCB Exit List Format and Contents . . . 221

Appendix E. Control Characters . . . 223
Machine Code . . . 223
ISO/ANSI/FIPS Control Characters . . . 225
Tables for Translating Codes . . . 226
Translating from EBCDIC to ASCII . . . 226
Translating from ASCII to EBCDIC . . . 226

Appendix F. Data Control Block Symbolic Field Names . . . 227
Data Control Block—Common Fields . . . 228
Data Control Block—BPAM, BSAM, QSAM . . . 229
Direct Access Storage Device Interface . . . 232
Magnetic Tape Interface . . . 232
Card Reader, Card Punch Interface . . . 233
Printer Interface . . . 233
Access Method Interface . . . 235
Data Control Block—ISAM . . . 238
Data Control Block—BDAM . . . 242

Appendix G. PDABD Symbolic Field Names 245

Abbreviations 247

Glossary 251

Index 259

Summary of Changes

Second Edition, June 1989

New Device Support

Track capacity determination formulas and examples for the 3380 models have been added to Appendix C, "Device Capacities."

Service Changes

Minor technical and editorial changes have been made.

First Edition, December 1988

New Programming Support for Release 1

The system can determine the block size for your data set. Information on system-determined block size has been added to the description of the DCB macro.

New Device Support for Release 1

Information has been added to Appendix C, "Device Capacities" on page 213 to reflect support of the following device types:

DASD

- IBM 3380 Direct Access Storage Models AJ4, BJ4, AK4, and BK4
- IBM 3380 Direct Access Storage Direct Channel Attach Model CJ2.

Storage Control

- IBM 3880 Storage Control Model 3 with 3380 AJ4/AK4 Attachment (feature 3005)
- IBM 3990 Storage Control Models 1 and 2

Cache Storage Control

- IBM 3880 Storage Control Model 23 with 3380 AJ4/AK4 Attachment (feature 3010)
- IBM 3990 Storage Control Model 3.

Information has also been added throughout the book to support the:

- IBM 4245, 4248, and 3262 Model 5 Printers.

Service Changes

Several new options have been added to the SETPRT macro.

MVS/DFP Version 3 publications have new order numbers. Publications listed in the preface reflect these new order numbers.

Minor technical and editorial changes have been made.

Preface

About This Book

This book is intended to help you use IBM data management access methods other than the virtual storage access method (VSAM) to process data sets. It contains general-use programming interfaces, which allow you to write programs that use the services of MVS/DFP.

This book discusses the non-VSAM data management macro instructions available in the assembler language. The macro instructions are presented in alphabetical 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. To learn how to write programs that create and process non-VSAM data sets, see:

- *MVS/ESA Data Administration Guide*, SC26-4505.

To learn about VSAM macros or to write programs that create and process VSAM data sets, see:

- *MVS/ESA Catalog Administration Guide*, SC26-4502, which describes how to create master and user catalogs
- *MVS/ESA Integrated Catalog Adminstration: Access Method Services Reference*, SC26-4500, and *MVS/ESA VSAM Catalog Administration: Access Method Services Reference*, SC26-4501, which describe the access method services commands used to manipulate VSAM data sets
- *MVS/ESA VSAM Administration Guide*, SC26-4518, which describes how to create VSAM data sets
- *MVS/ESA VSAM Administration: Macro Instruction Reference*, SC26-4517, which describes how to code the macro instructions required with VSAM data sets.

The following access methods and macros are shown in this publication for the sake of compatibility only. Although they are still supported, their use is not recommended, and where applicable, alternatives are suggested.

- The access methods BDAM and EXCP—we recommend you use VSAM key-sequenced data sets (KSDS) instead of BDAM.
- The access methods BISAM and QISAM—we recommend you use VSAM instead.
- The macros:
  - DCB (BDAM, BISAM, QISAM)
  - ESETL
  - FREEDBUF
  - GET (QISAM)
  - PUT (QISAM)
  - READ (BDAM, BISAM)

- RELEX
- SETL
- WRITE (BDAM, BISAM)

Required Product Knowledge

To use this book effectively, you should be familiar with:

- assembler language
- job control language.

If you know how to write assembler language programs and use job control statements, you can use this book and *MVS/ESA Data Administration Guide*, SC26-4505, to write programs that create and process data sets.

Required Publications

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

Publication Title	Order Number
Assembler H Version 2 Application Programming: Guide	SC26-4036
Assembler H Version 2 Application Programming: Language Reference	SC26-4037
MVS/ESA Application Development Guide	GC28-1821
MVS/ESA Application Development Macro Reference	GC28-1822
MVS/ESA Data Administration Guide	SC26-4505
MVS/ESA JCL Reference	GC28-1829
MVS/ESA JCL User's Guide	GC28-1830

Related Publications

Some publications from the MVS/SP Version 3 library are referenced in this book. The *MVS/ESA Library Guide for System Product Version 3*, GC28-1563, contains a complete listing of the MVS/SP Version 3 publications and their counterparts for the prior version.

The *MVS/ESA Data Facility Product Version 3: Master Index*, GC26-4512, contains both an index to the MVS/DFP library and a summary of the changes made to the library. You can use it to:

- Find information in other MVS/DFP publications
- Determine how new programming support changes information in the MVS/DFP library
- Determine which MVS/DFP publications have been changed.

In addition, the following publications may be helpful:

Publication Title	Order Number
MVS/ESA Data Facility Product Version 3: Diagnosis Guide	LY27-9550
MVS/ESA Message Library: System Messages Volume 1	GC28-1812
MVS/ESA Message Library: System Messages Volume 2	GC28-1813
MVS Storage Management Library: Configuring Storage Subsystems	SC26-4409

Referenced Publications

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

Short Title	Publication Title	Order Number
Application Development Guide	*MVS/ESA Application Development Guide*	GC28-1821
Application Development Macro Reference	*MVS/ESA Application Development Macro Reference*	GC28-1822
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*	SC26-4037
Checkpoint/Restart User's Guide	*MVS/ESA Checkpoint/Restart User's Guide*	SC26-4503
Data Administration Guide	*MVS/ESA Data Administration Guide*	SC26-4505
Data Areas Volume 5	*MVS/ESA Diagnosis: Data Areas Volume 5*	LY28-1047
DFP: Customization	*MVS/ESA Data Facility Product Version 3: Customization*	SC26-4504
DFP: Diagnosis Reference	*MVS/ESA Data Facility Product Version 3: Diagnosis Reference*	LY27-9551
IBM System/370 XA Principles of Operation	*IBM System/370 Extended Architecure Principles of Operation*	SA22-7085
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

Short Title	Publication Title	Order Number
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	*IBM 4245 Printer Model 1 Component Description and Operator*	GA33-1541
IBM 4248 Printer Model 1 Description	*IBM 4248 Printer Model 1 Description*	GA24-3927
JCL Reference	*MVS/ESA JCL Reference*	GC28-1829
JCL User's Guide	*MVS/ESA JCL User's Guide*	GC28-1830
Magnetic Tape Labels and File Structure	*MVS/ESA Magnetic Tape Labels and File Structure Administration*	SC26-4511
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	*Resource Access Control Facility (RACF) General Information*	GC28-0722
SPL: Application Development Guide	*MVS/ESA System Programming Library: Application Development Guide*	GC28-1852
System Codes	*MVS/ESA Message Library: System Codes*	GC28-1815
System—Data Administration	*MVS/ESA System—Data Administration*	SC26-4515
Utilities	*MVS/ESA Data Administration: Utilities*	SC26-4516
VSAM Administration Guide	*MVS/ESA VSAM Administration Guide*	SC26-4518
VSAM Administration: Macro Instruction Reference	*MVS/ESA VSAM Administration: Macro Instruction Reference*	SC26-4517
3380 DAS Reference Summary	*IBM 3380 Direct Access Storage Reference Summary*	GX26-1678

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.

Before coding programs that request supervisor services, familiarize yourself with the information contained in *Application Development Guide.* This book explains how to write assembler language programs that use MVS services available to all programs. *Application Development Macro Reference*, a companion volume to *Application Development Guide*, describes the macros available to all application programs. If you want to use VSAM (virtual storage access method), see the VSAM publications listed in the preface.

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 211 lists the macro instructions available for each access method.

24- and 31-Bit Addressing Considerations

Unless otherwise stated, executable macros described in this book can be executed only in 24-bit addressing mode, and data referenced by the macros must reside below the 16M line.

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*.

Notational Conventions

A uniform system of notation describes the format of data management macro instructions. This notation is not part of the language; it merely provides a basis for describing the structure of the macros.

The command format illustrations in this book use the following conventions:

Brackets

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

Examples:

- [length]
- [MF=E]

Braces

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

Examples:

- BFTEK={S|A}
- {K|D}
- {address|S|O}

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.

OR Sign

Items separated by a vertical bar (|) represent alternative items. No more than one of the items may be selected.

Examples:

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

Ellipses

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

Example:

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

Other Punctuation

Other punctuation (parentheses, commas, etc.) must be entered as shown.

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 Bold

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

Examples:

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

Italics

Italics type are used for elements for which you code values that you choose, usually according to specifications and limits described for each parameter. *Italics* type also specifies fields to be supplied by the user.

Examples:

- number
- image-id
- count

' '

A ' ' in the macro format indicates that a blank (an empty space) must be present before the next parameter.

( )

Parentheses **( )** must enclose subfields if more than one is specified. If only one subfield is specified, you may omit the parentheses.

Blank Symbol

The blank symbol, b, indicates omitted operands.

Example:

b	**PDABD**	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, to distinguish the register number from an A-type address. For example, if you code register 3, use the form **(3)**. The following is an example with the CLOSE macro:

```
CLOSE   ((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, you can use general register 1 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, you can use general register 0 as an operand. The register can be specified as a decimal digit 0 enclosed in parentheses. When register 0 is used as an operand, the instruction that loads the parameter value into the register is not included in the macro expansion.

RX-Type Address

When this notation is shown, you can specify the operand 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.

A-Type Address
: When this notation is shown, you can specify the operand as any address that can be written as a valid assembler-language A-type address constant. You can write an A-type address constant 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 *Assembler H V2 Application Programming: Language Reference*.

absexp
: 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 *Assembler H V2 Application Programming: Language Reference*.

relexp
: 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 *Assembler H V2 Application Programming: Language Reference*.

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 the macro expansion can include a branch around the in-line parameter list, if present, and list 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.

Some macro instructions 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 24-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, an efficient way to clear the high-order byte of a register is to code 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. Comments may be continued after column 16.

Note that if column 72 is filled in on one line and you try to continue an operand or start a new statement after column 16 on the next line, this statement will be taken as a comment belonging to the previous statement.

You can code the operand field being continued in one of two ways. 1) The operand field can be coded through column 71, with no blanks, and be continued in column 16 of the next line, or 2) 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 is shown in the following illustration:

```
Name      Operation  Operand                 Comments          Cont'd
MYFILE    DCB        DSORG=PS,               THIS IS ONE WAY   X
                     DDNAME=IP,                                X
                     EODAD=EOFRTN,                             X
                     MACRF=GM,RECFM=F,                         X
                     LRECL=80,                                 X
                     BLKSIZE=80
```

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

Macro Instruction Descriptions

BLDL—Build a Directory Entry List (BPAM)

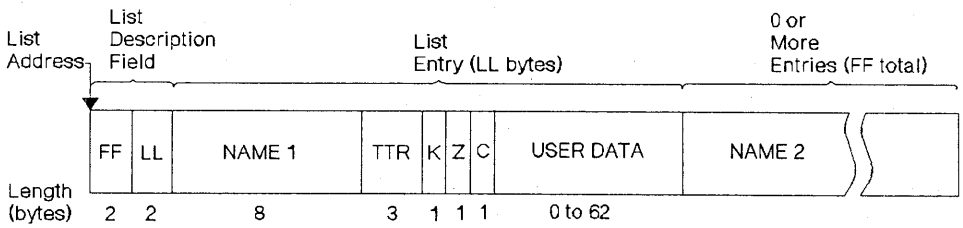
The BLDL macro is used to obtain a list of information from the directory of a partitioned data set. The problem program must supply a storage area that must include information about the number of entries in the list, the length of each entry, and the name of each data set member (or alias) before the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

The BLDL macro is written:

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

dcb address—RX-Type Address, (2-12) or (1)
specifies the address of the data control block for an open partitioned data set, or you can specify zero 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)
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 bytes to fill in the **Z** and **C** fields. If the **LL** field is 12 bytes, 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 backspaces the current volume one data block (physical record). All input and output operations must be tested for completion before the BSP macro is issued. Do not use the BSP macro if the CNTRL, NOTE, or POINT macro is being used.

Any attempt to backspace across a file mark results in a return code of X'04' and your tape or direct access volume is not 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)
specifies the address of the data control block for the volume to be backspaced. You must open the data set on the volume to be backspaced 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')	The supplied DCB or its DEB is invalid.
04 (X'04')	07 (X'07')	Backspace detected an invalid extent 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)

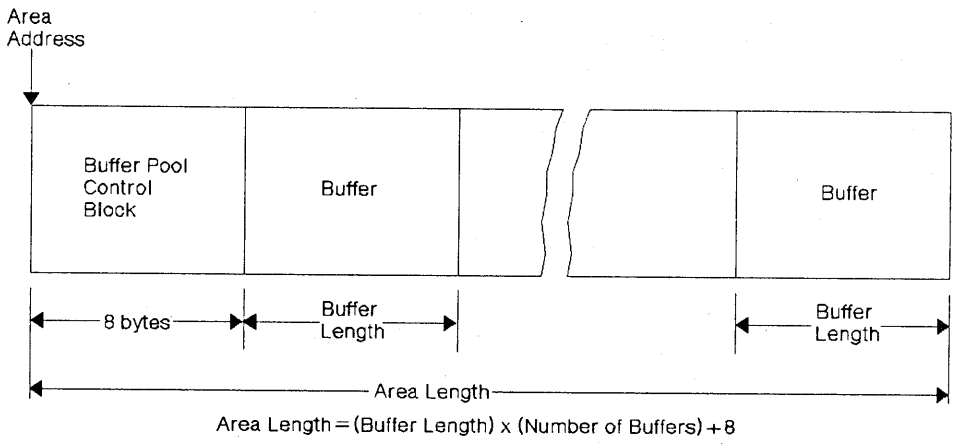
The BUILD macro is used to construct a buffer pool in an area provided by the problem program. The buffer pool may be used by more than one data set through separate data control blocks. Individual buffers are obtained from the buffer pool using the GETBUF macro, and buffers are returned to the buffer 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)
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:

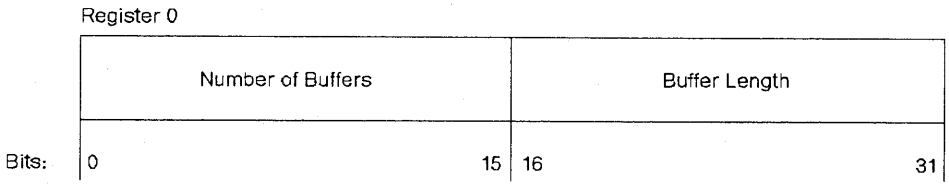


Area Length = (Buffer Length) x (Number of Buffers) + 8

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

buffer length—symbol, decimal digit, absexp, or (2-12)
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 constructs a buffer pool and a record area 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. Using this macro before the data set is opened, or before the end of the DCB open exit routine, provides 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. You cannot specify the BUILDRCD macro when logical records exceed 32760 bytes.

It is your responsibility to release the buffer pool and the record area after issuing a CLOSE macro for all the data control blocks that use the buffer pool and the record area.

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)
specifies the address of the area to be used as a buffer pool. The area must start on a fullword boundary.

Note: area length = [(buffer length) × (number of buffers) + 12]

number of buffers—symbol, decimal digit, absexp, or (2-12)
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)
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.

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

It is your responsibility to release the buffer pool and the record area after issuing a CLOSE macro 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
specifies that the BUILDRCD macro instruction is used to create a parameter list that is referenced by an execute form instruction.

Note: You can construct a parameter list 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, code the operands 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***)**

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***)**
specifies that the execute form of the BUILDRCD macro instruction is used, and an existing parameter list (created by a list-form instruction) is 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 places the active task 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, you must issue a CHECK macro for each input and output operation. The CHECK macros must be issued in the same order as the READ or WRITE macros were issued for the data set. For BDAM or BISAM, you can use either a CHECK or a WAIT macro. For information on when you can use the WAIT macro, 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 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, if the record format is **F**, **FB**, **D**, **DB**, or **U**. Translation occurs when 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)
specifies the address of the data event control block created or used by the associated READ or WRITE macro.

DSORG={IS|ALL}

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 creates output data set labels and allows you to position volumes. 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. You can specify final volume positioning or disposition for the current volume 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 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 User's Guide*.)

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 is 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 rewound and unloaded, if necessary.

If **FREE=CLOSE** has been coded in the DD statement associated with this data set, coding the **DISP** option in the CLOSE macro results 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 used only 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 for 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 before the normal CLOSE (without **TYPE=T**) and after the temporary close was a write, then any unused space is released.

MODE=24|31

You can code CLOSE **MODE=31** to specify a long form parameter list that can contain 31-bit addresses. The default, **MODE=24**, specifies a standard form parameter list with 24-bit addresses. Your program does not need to be executing in 31-bit addressing mode to use **MODE=31** in the CLOSE macro. This parameter specifies the form of the parameter list, not the addressing mode of the program.

The standard form parameter list must reside below 16M, but the calling program may be above 16M. Assume that all ACBs and DCBs are below 16M.

The long form parameter list can reside above or below 16M. 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 is issued and the data set is not 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.

For additional information and coding restrictions, see *Data Administration Guide*.

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. You can construct a maximum length list 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
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
specifies that the CLOSE macro instruction is used to create a data management parameter list that is referred to by an execute-form instruction.

MODE=24|31
You can code CLOSE **MODE=31** to specify a long form parameter list that can contain 31-bit addresses. The default, **MODE=24**, specifies a standard form parameter list with 24-bit addresses. Your program does not need to be executing in 31-bit addressing mode to use **MODE=31** in the CLOSE macro. This parameter specifies the form of the parameter list, not the addressing mode of the program.

The standard form parameter list must reside below 16M, but the calling program may be above 16M. Assume that all ACBs and DCBs are below 16M.

The long form parameter list can reside above or below 16M. 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 is issued and the data set is not 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***)** [,**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 is used.

TYPE=T—Same as standard form.

MF=(E,*address of the list form***)**
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 can contain 31-bit addresses. The default, **MODE=24**, specifies a standard form parameter list with 24-bit addresses. Your program does not need to be executing in 31-bit addressing mode to use **MODE=31** in the CLOSE macro. This parameter specifies the form of the parameter list, not the addressing mode of the program.

The standard form parameter list must reside below 16M, but the calling program may be above 16M. Assume that all ACBs and DCBs are below 16M.

The long form parameter list can reside above or below 16M. 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 is issued and the data set is not 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.

Return Codes from CLOSE

When your program receives control after it has issued a CLOSE macro, a return code in register 15 indicates whether all data sets were closed successfully:

Return Code (15)	Meaning
0(0)	All data sets were closed successfully.
4(4)	At least one data set (VSAM or non-VSAM) was not closed successfully.

CNTRL—Control Online Input/Output Device (BSAM and QSAM)

The CNTRL macro controls magnetic tape drives (BSAM only for a data set that is not open for output), online 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, issue a CNTRL macro after every input request. For blocked records, issue a CNTRL macro 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. You cannot use a CNTRL macro 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, you can use the CNTRL macro 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: Do not use the CNTRL macro 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 *Programming Support for the IBM 3505 Card Reader and the 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
specifies the address of the data control block for the data set opened for the online device.

SS,{1|2}
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}
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}
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
indicates that the control function requested is to backspace the magnetic tape past a tape mark, then forward space over the tape mark.

FSM
indicates that the control function requested is to forward space the magnetic tape over a tape mark, then backspace past the tape mark.

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

FSR
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)
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. For more information on register contents when control is passed to the error analysis routine, see *DFP: Customization*. If a tape mark 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)

Use of the DCB (BDAM) macro is not recommended; we recommend you use a device-independent access method such as BSAM, BPAM, or QSAM instead.

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

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.

Note: When creating a DCB to open a data set that has been allocated to an SMS-managed volume, do not specify values that would change the data set to a type which cannot be SMS-managed, such as **DSORG**=**DAU**.

When you create or process a BDAM data set, you can specify the following operands in the DCB macro instruction:

BFALN=**{F**\|**D}**

specifies the boundary alignment for each buffer in the buffer pool. You can specify the **BFALN** operand 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 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 you use the BUILD macro instruction 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

specifies that the data set is being created for or contains variable-length spanned records. You can code the **BFTEK=R** operand 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=*absexp* (maximum value is 32760)

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 you can specify 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 213. For additional information about space allocation, see *Data Administration Guide*.

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. Block size can also be derived from the JCL keyword LIKE. For more information on LIKE, see *JCL User's Guide*.

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

If the buffer pool is constructed automatically, dynamically, or by a GETPOOL macro instruction, you do not need to use the **BUFCB** operand because the system places the address of the buffer pool control block into the data control block. Also, if the problem program is to control all buffering, omit the **BUFCB** operand.

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)
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.

You must specify the **BUFL** operand when processing an existing BDAM data set with dynamic buffering. 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.

You can omit the **BUFL** operand 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)
specifies the number of buffers to be constructed by a BUILD macro instruction, or the number of buffers and/or segment work areas to be acquired automatically 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, you must specify the number of buffers 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, you can omit the **BUFNO** operand unless you need it 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*

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**}

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 characters can be specified:

DA

specifies a direct organization data set.

DAU

specifies a direct organization data set that contains location-dependent information that would make it unmovable.

Note: A **DSORG**=**DAU** data set cannot be SMS-managed.

When a BDAM data set is created, the basic sequential access method (BSAM) is used. You must code the **DSORG** operand in the DCB macro instruction as **DSORG**=**PS** or **PSU** when the data set is created, and code the **DCB** subparameter in the corresponding DD statement 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*

specifies the address of the problem program exit list. The **EXLST** operand is required 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 221. For additional information about exit list processing, see *DFP: Customization*.

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)
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.

Key length can be derived from the data class associated with the data set. Key length can also be derived from the JCL keyword LIKE. However, if **KEYLEN** is specified in the DCB macro instruction, it overrides the value derived from data class or LIKE. For more information, see *JCL User's Guide*.

LIMCT=*absexp*
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**])}}

specifies the type of macro instructions (READ, WRITE, CHECK, and WAIT) that are used to process the data set. 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 characters can be coded for BDAM:

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

C
specifies that the CHECK macro instruction is 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 to be 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]}

specifies the optional services 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. You may code the following characters in any order, in any combination, and without commas 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 prevents the open routines from merging any of the other OPTCD bits from the format-1 DSCB in the DCB. Both methods update the OPTCD bits 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**]}
specifies the record format and characteristics of the data set being created or processed. 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). **Note:** Track overflow is not supported on DASD models 3375 through 3380.

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.

Record format can be derived from the data class associated with the data set. Record format can also be derived from the JCL keyword LIKE. However, if **RECFM** is specified in the DCB macro instruction, it overrides the value derived from data class or LIKE. For more information, see *JCL User's Guide*.

SYNAD=*relexp*
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 *DFP: Customization*.

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)

Use of the DCB (BISAM) macro is not recommended; we recommend you use VSAM instead.

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

Note: You cannot use a BISAM DCB to open a data set allocated to an SMS-managed volume.

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.

When you create or process a BISAM data set, you can specify the following operands in the DCB macro instruction:

BFALN=**{F**|**D}**

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 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.

BUFCB=*relexp*

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

You can omit the **BUFCB** operand if you request dynamic buffering or use the GETPOOL macro instruction to construct the buffer pool, because the system places the address of the buffer pool control block into the data control block. Also, if the problem program is to control all buffering, omit the **BUFCB** operand.

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)

specifies the length, in bytes, of each buffer in the buffer pool to be 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 **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 *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)

specifies the number of buffers requested for use with dynamic buffering, or 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=*symbol*

specifies the name used to identify the job control language data definition (DD) statement that defines the ISAM 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

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=*relexp*

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 221. For additional information about exit list processing, see *DFP: Customization*.

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])**}}

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 characters can be coded for BISAM:

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 specified, WAIT macro instructions must be used to test for completion of I/O operations.

R

specifies that READ macro instructions are to be used.

S

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

U

specifies that records in the data set are to 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*

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*

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 significant 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)

specifies the maximum number of READ and 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 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**])}

specifies the optional services performed by the control program when creating or updating an ISAM data set. You must request all optional services 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. You may code the following characters in any order, in any combination, and without commas 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 statistics in certain 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 you use the **OPTCD** operand, you must specify **OPTCD=R** to get the reorganization statistics.

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

SMSI=*absexp* (maximum value is 65535)
specifies the length, in bytes, required to contain the highest-level master index for the data set being processed. Look at the DCBNCRHI field of the data control block to determine the size required. 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 significant 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=*absexp* (maximum value is 65535)
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. Code the **SMSW** operand together with the **MSWA** operand. If you code the **SMSW** operand but the size you specified is less than the minimum required, the task is abnormally terminated. *Data Administration Guide* 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 significant 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=*relexp*
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 *DFP: Customization*.

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 a basic partitioned access method (BPAM) data set is constructed during assembly of the problem program. You must code the **DSORG** and **MACRF** operands 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.

Note: When creating a DCB to open a data set that has been allocated to an SMS-managed volume, do not specify values that would change the data set to a type which cannot be SMS-managed, such as **DSORG=POU**. Refer to *Data Administration Guide* for further information.

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

BFALN={**F**|**D**}
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)
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 you can specify 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 213. For additional information about space allocation, see *Data Administration Guide*.

For fixed-length records, the value specified in the **BLKSIZE** operand should be a multiple of the value specified for the logical record length (LRECL).

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. Block size can also be derived from the JCL keyword LIKE. For more information on LIKE, see *JCL User's Guide*.

System-Determined Block Size: For blocked DASD data sets, if the block size is not specified at the time that the data set is created, and the **LRECL** and **RECFM** are known, the system derives an optimum block size for the data set. This system-determined block size is retained in the data set label. When the data set is opened for output, OPEN checks the block size in the data set label. If it is a system-determined block size, and the **LRECL** or **RECFM** have changed from those specified in the data set label, OPEN will rederive an optimum block size for the data set.

BUFCB=*relexp*
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, you can omit the **BUFCB** operand because the system places

the address of the buffer pool control block into the data control block. Also, if the problem program is to control all buffering, omit the **BUFCB** operand.

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)
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, specify the **BUFL** operand.

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)
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*
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**}
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 that would make it unmovable.

Notes:

1. Unmovable data sets cannot be SMS-managed.
2. If BSAM or QSAM is used to add or retrieve a single member of a partitioned data set, specify **DSORG=PS** or **PSU** in the BSAM or QSAM DCB. 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*
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 but no **EODAD** address was supplied, the task is abnormally terminated. For additional information on the EODAD routine, see *Data Administration Guide* and *DFP: Customization*.

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*
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 221. For additional information about exit list processing, see *DFP: Customization*.

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=*absexp* (maximum value is 255)
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 is ignored if specified in the DD statement.

Key length can be derived from the data class associated with the data set. Key length can also be derived from the JCL keyword LIKE. However, if **KEYLEN** is specified in the DCB macro instruction, it overrides the value derived from data class or LIKE. For more information, see *JCL User's Guide*.

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

For fixed-length records, if the records are unblocked, the value specified in the **LRECL** operand must equal the value specified in the **BLKSIZE** operand. For variable-length records, 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.

Record length can be derived from the data class associated with the data set. Record length can also be derived from the JCL keyword LIKE. For undefined-length records, if **LRECL** is specified in the DCB macro instruction, it overrides the value derived from data class or LIKE. For more information, see *JCL User's Guide*.

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

R
specifies that READ macro instructions are to be used. This operand automatically allows you 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 allows you 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 is to be used.

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

NCP=*absexp* (maximum value is 99)
specifies the maximum number of READ and WRITE macro instructions that are issued before the first CHECK macro instruction is issued to test 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 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**]}}
specifies the optional services performed by the system.

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. (See *Mass Storage System (MSS) Extensions Services: Reference* for more information on MSS. Use of MSS is not recommended.

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**]}}
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 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 223.

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 223.

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).

Note: Track overflow is not supported on DASD models 3375 through 3380.

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.

Record format can be derived from the data class associated with the data set. Record format can also be derived from the JCL keyword LIKE. However, if **RECFM** is specified in the DCB macro instruction, it overrides the value derived from data class or LIKE. For more information, see *JCL User's Guide*.

SYNAD=*relexp*

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 *DFP: Customization*.

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 in this manner, 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 a basic sequential access method (BSAM) data set is constructed during assembly of the problem program. You must code the **DSORG** and **MACRF** operands 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 that can supply the operands.

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][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}[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]|[A][M]} {D[B][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.

Note: When creating a DCB to open a data set that has been allocated to an SMS-managed volume, do not specify values that would change the data set to a type which cannot be SMS-managed, such as **DSORG=PSU**.

When you create or process a BSAM data set, you can specify the following operands in the DCB macro instruction:

BFALN={F|D}
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 from the same source.

BFTEK=R
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)
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 you can specify in the **BLKSIZE** operand depends on the device type and the record format being used. Device capacity for direct access devices is described in Appendix C, "Device Capacities" on page 213. For additional information about device capacity, see the relevant device publication.

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 **BLKSIZE** value is 18 bytes. The maximum value 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 BSAM is used to read variable-length spanned records the value specified for the **BLKSIZE** operand must be as large as the longest possible record segment in the 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. Block size can also be derived from the JCL keyword LIKE. For more information on LIKE, see *JCL User's Guide*.

System-Determined Block Size: For blocked DASD data sets, if the block size is not specified at the time that the data set is created, and the **LRECL** and **RECFM** are known, the system derives an optimum block size for the data set. This system-determined block size is retained in the data set label. When the data set is opened for output, OPEN checks the block size in the data set label. If it is a system-determined block size, and the **LRECL** or **RECFM** have changed from those specified in the data set label, OPEN will rederive an optimum block size for the 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*
specifies the address of the buffer pool control block when a buffer pool is constructed by a BUILD macro instruction.

If the buffer pool is to be constructed automatically or by a GETPOOL macro instruction, omit the **BUFCB** operand. This is because the system places the address of the buffer pool control block into the data control block. Also, if the problem program is to control all buffering, omit the **BUFCB** operand.

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)
specifies the length, in bytes, for each buffer in the buffer pool when the buffer pool is acquired automatically. 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. If the problem program requires larger buffers, the **BUFL** operand is required. 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)
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, omit the **BUFNO** operand.

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**}
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*
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**}
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 you code **DEVD=DA** in a DCB macro instruction (or omit the **DEVD** operand, which causes a default to **DA**), you can use later the data control block constructed during assembly for any of the other devices, but, if you code **DEVD=RD**, you can use the data control block 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. Also, 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 you code **DEVD=PR**, **PC**, or **RD**, do not code the DCB macro 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*
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**}
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}

These values specify the recording technique for 7-track tape. One of the above four values can be coded. If the **TRTCH** operand is omitted, odd parity with no translation or conversion is assumed. The values 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.

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

DEVD=PR
[,PRTSP={0|1|2|3}]

specifies that the data control block is used for an online 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}

specifies the line spacing on the printer. This operand is not valid if the **RECFM** operand specifies either machine (**RECFM=M**) or 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][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 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]

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.

R

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

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

STACK={1|2}

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]}

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|E][O|R]
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}

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**]}

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**}

specifies the data set organization and whether the data set contains any location-dependent information that would make it unmovable. The following characters 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.

Notes:

1. Unmovable data sets cannot be SMS-managed.

Source: You must code the **DSORG** operand in the DCB macro instruction.

EODAD=*relexp*

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 was supplied to the data control block, the task is abnormally terminated. For additional information on the EODAD routine, see *DFP: Customization.*

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*

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 routines, end-of-volume, block count exits, defining a forms control buffer (FCB) image, using the JFCBE

exit (for the IBM 3800 Printing Subsystem), 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 221. For additional information about exit list processing, see *DFP: Customization*.

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)
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 is ignored if specified in the DD statement.

Key length can be derived from the data class associated with the data set. Key length can also be derived from the JCL keyword LIKE. However, if **KEYLEN** is specified in the DCB macro instruction, it overrides the value derived from data class or LIKE. For more information, see *JCL User's Guide*.

LRECL={*absexp*|**X**}
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 in the **LRECL** operand cannot exceed the value specified in 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 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. However, the **LRECL** operand is not checked for validity.

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

For undefined-length records, omit the **LRECL** operand; the actual length is 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.

X

When using BSAM 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). Specify **LRECL=X** if the logical record length is greater than 32756 bytes.

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.

Record length can be derived from the data class associated with the data set. Record length can also be derived from the JCL keyword LIKE. However, if **LRECL** is specified in the DCB macro instruction, it overrides the value derived from data class or LIKE. For more information, see *JCL User's Guide*.

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

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 combinations shown above. The following characters can be coded for BSAM:

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 allows you to use NOTE macro instructions with the data set. Do not code **P** 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)

specifies the maximum number of READ and WRITE macro instructions that are 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**}}

specifies the optional services 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, you must omit the **OPTCD** operand from the DCB macro instruction if either of these options is requested in a DD statement.

You may code the following characters in any order, in any combination, and without commas between characters.

C

specifies that chained scheduling is 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 is to 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 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 *MVS/ESA 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

For printers, **U** is specified 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. (See *Mass Storage System (MSS) Extensions Services: Reference* for more information on MSS.)

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 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**]|[**A**][**M**]}
{**D**[**B**][**S**][**A**]}
{**F**[**B**|**S**|**T**|**BS**|**BT**][**A**|**M**]}}

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 characters 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 223.

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 223. **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; except for 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 fixed-length 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(s) (if required).

Note: Track overflow is not supported on DASD models 3375 through 3380.

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.

Record format can be derived from the data class associated with the data set. Record format can also be derived from the JCL keyword LIKE. However, if **RECFM** is specified in the DCB macro instruction, it overrides the value derived from data class or LIKE. For more information, see *JCL User's Guide*.

SYNAD=*relexp*
specifies the address of the error analysis (SYNAD) 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 *DFP: Customization*.

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 in this manner, 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 IBM 3800 Model 3 using all-points addressability, the SYNAD routine is 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)

Use of the DCB (QISAM) macro is not recommended; we recommend you use VSAM instead.

The data control block for a queued indexed sequential access method (QISAM) data set is constructed during assembly of the problem program. You must code the **DSORG** and **MACRF** operands in the DCB macro instruction, but the other DCB operands can be supplied to the data control block 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.

Note: You cannot use a QISAM DCB to open a data set allocated to an SMS-managed volume.

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.

When you create or process a QISAM data set, you can specify the following operands in the DCB macro instruction:

BFALN={**F**|**D**}
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. 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, 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)
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. You must specify the **BLKSIZE** operand when creating an ISAM data set. When processing an existing ISAM data set, you must omit the **BLKSIZE** operand (it is supplied by the data set label).

You need to consider the track capacity of the direct access device being used when specifying the block size for an ISAM data set. 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 213. For additional information about space allocation, see *Data Administration Guide*.

If fixed-length records are used, the value specified in the **BLKSIZE** operand must be a whole number 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*
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, omit the **BUFCB** operand, because the system places the address of the buffer pool control block into the data control block.

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)
specifies the length, in bytes, of each buffer in the buffer pool to be 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, because 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)
specifies the number of buffers to be constructed by a BUILD macro instruction, or 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)
specifies the number of tracks on each cylinder that is reserved as an overflow area. The overflow area contains 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*
specifies the name used to identify the job control language data definition (DD) statement that defines the ISAM 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**}
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. You can specify **ISU** only when creating an ISAM data set.

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*
specifies the address of the routine to be given control when the end of an input data set is reached. For ISAM, this operand applies 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* and *DFP: Customization.*

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*
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 221. For additional information about exit list processing, see *DFP: Customization.*

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

KEYLEN=*absexp* (maximum value is 255)
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=*absexp* (maximum value is device-dependent)
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**).

You need to consider the track capacity of the direct access device being used when using maximum-length logical records. 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 213. For additional information about space allocation, see *Data Administration Guide*.

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**])}}

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 characters can be coded for QISAM:

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=*absexp* (maximum value is 99)

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**]}

specifies the optional services performed by the system when an ISAM data set is being created or updated. You may code the following characters in any order, in any combination, and without commas 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 is to 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 instruc-

tion), 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 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 you use the **OPTCD** operand, you must specify **OPTCD=R** 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 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]}

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*

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. Do not specify the delete option (**OPTCD=L**) 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*

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 *DFP: Customization*.

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 a queued sequential access method (QSAM) data set is constructed during assembly of the problem program. You must code the **DSORG** and **MACRF** operands 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 that can supply the operand.

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**|*nnnnn***K**}] ,**MACRF**={{(**G**{**M**|**L**|**D**}[**C**])} {(**P**{**M**|**L**|**D**}[**C**])} {(**G**{**M**|**L**|**D**}[**C**],**P**{**M**|**L**|**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**]|[**A**][**M**]} {**D**[**B**][**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.

Notes:

1. When creating a DCB to open a data set that has been allocated to an SMS-managed volume, do not specify values that would change the data set to a type which cannot be SMS-managed, such as **DSORG=PSU**.
2. For information on additional operands for the DCB macro for the IBM 3890 Document Processor, see *IBM 3890 Document Processor Machine and Programming Description*.

When you create or process a QSAM data set, you can specify the following operands in the DCB macro instruction:

BFALN={F|D}
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, see the description of the **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, 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}
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 you can specify:

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 buffering technique efficiently, you 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 bytes for IBM standard labels) 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.

The actual value that you can specify 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 213. (For additional information about device capacity, refer to the relevant device publication.)

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 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 213, 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 a whole number 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** value 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. Block size can also be derived from the JCL keyword LIKE. For more information on LIKE, see *JCL User's Guide*.

System-Determined Block Size: For blocked DASD data sets, if the block size is not specified at the time that the data set is created, and the pk.LRECL and **RECFM** are known, the system derives an optimum block size for the data set. This system-determined block size is retained in the data set label. When the data set is opened for output, OPEN checks the block size in the data set label. If it is a system-determined block size, and the **LRECL** or **RECFM** have changed from those specified in the data set label, OPEN will rederive an optimum block size for the 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*
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, you can omit the **BUFCB** operand, because the system places the address of the buffer pool control block into the data control block.

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)
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)
specifies the number of buffers in the buffer pool constructed by a BUILD or BUILDRCD macro instruction or 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**}
specifies the length, in bytes, of the block prefix used with an ISCII/ASCII tape data set. When QSAM is used to read an ISCII/ASCII tape data set, 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*
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*]
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 you code **DEVD=DA** in a DCB macro instruction (or omit the **DEVD** operand, which causes a default to **DA**), you can use later the data control block constructed during assembly for any of the other devices, but, if you code **DEVD=RD**, you can use the data control block 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 you code **DEVD=PR**, **PC**, or **RD**, do not code the DCB macro 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}
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}
These values specify the recording technique for 7-track tape. One of the above four values can be coded. If the **TRTCH** operand is omitted, odd parity with no translation or conversion is assumed. The values 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, but no data-conversion feature.

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

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

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

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).

Note: You cannot use MODE and FUNC subparameters with this specification.

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

R

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

STACK={1|2}

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]}

defines the type of 3525 card punch data sets that are 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 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.

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]

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 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}

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 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**]}

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 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}
specifies the data set organization and whether the data set contains any location-dependent information that would make it unmovable. The following characters 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.

Note:

An unmovable data set cannot be SMS-managed.

Source: You must code the **DSORG** operand in the DCB macro instruction.

EODAD=*relexp*
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 a file mark is read or when more data is requested after reading a truncated block. If the end of the data set is reached but no **EODAD** address was 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 *Data Administration Guide* and *DFP: Customization*.

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|ABE}
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. You can specify this action when opening a data set 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 releases the buffer associated with the data block. You can specify this action when opening a data set for INPUT, RDBACK, or UPDAT.

ABE

specifies that the error is to result in the abnormal termination of the task. You can specify this action when opening the data set 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, do not issue FEOV 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*

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 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 221. For additional information about exit routine processing, see *DFP: Customization*.

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**}

specifies the length, in bytes, for fixed-length records, or it specifies the maximum length, in bytes, for variable-length or undefined-length (output only) 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 bytes) 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 else the maximum-length record is 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), you must specify **LRECL**=**X** and use GET or PUT locate mode.

For ISO/ANSI/FIPS variable-length spanned records (**RECFM**=**DS** or **RECFM**=**DBS**), you may use the extended logical record interface (XLRI)

when the maximum logical record length exceeds 32760 bytes. XLRI must be invoked by specifying **LRECL=0K** or **LRECL=nnnnnK**.

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.

LRECL=0K
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**.

X
Specify if the logical record length exceeds the maximum block size (32760 bytes), and use GET or PUT locate mode.

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 indicates 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 indicates '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.

Record length can be derived from the data class associated with the data set. Record length can also be derived from the JCL keyword LIKE. However, if **LRECL** is specified in the DCB macro instruction, it overrides the value derived from data class or LIKE. For more information, see *JCL User's Guide*.

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**|**D**}[**C**])}}
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 characters 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**}}

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, you must omit the **OPTCD** operand from the DCB macro instruction if either of these options is requested in a DD statement.

You may code the following characters in any order without commas between characters:

C

specifies that chained scheduling is 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 is to 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 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 after 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 more information about ISCII/ASCII to EBCDIC or EBCDIC to ISCII/ASCII translations, see *Magnetic Tape Labels and File Structure*.

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

For printers, **U** is specified for a printer with the universal-character-set feature (UCS) 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 3480 Magnetic Tape Subsystem, sets to write-tape-immediate mode.

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. (See *Mass Storage System (MSS) Extensions Services: Reference* for more information on MSS.)

W

specifies, for DASD, 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 treats EOF labels as EOV labels until the volume serial list is exhausted. After all the volumes have been read, control is passed to your 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.

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**]|[**A**][**M**]}
{**D**[**B**][**S**][**A**]}
{**F**[**B**|**S**|**T**|**BS**|**BT**][**A**|**M**]}}

specifies the record format and characteristics of the data set being created or processed. All record formats can be used in QSAM. The following characters 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 223.

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 223. **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; except for the last block or track in the data set, the data set does not contain any truncated blocks or unfilled tracks. Do not code **S** to retrieve fixed-length 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).

Note: Track overflow is not supported on DASD models 3375 through 3380.

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.

Record format can be derived from the data class associated with the data set. Record format can also be derived from the JCL keyword LIKE. However, if **RECFM** is specified in the DCB macro instruction, it overrides the value derived from data class or LIKE. For more information, see *JCL User's Guide*.

SYNAD=*relexp*

specifies the address of the error analysis (SYNAD) 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 *DFP: Customization*.

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 generates a dummy control section that provides symbolic names for the fields in one or more data control blocks. The DCBD macro maps the assembler version of the DCB. 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 227. Attributes of the symbolically named fields in the dummy section are the same as the fields in the data control blocks, except for 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 your user program. The macro labels are structured as DCBxxxxx, 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). You can issue the DCBD macro instruction only once within any assembled module; however, you can use the resulting symbolic names for any number of data control blocks by changing the address in the dummy section base register. You can code the DCBD macro instruction at any point in a control section. However, if it is coded at any point other than at the end of a control section, you must code a CSECT instruction to resume the control section.

The DCBD macro is written:

b	**DCBD**	[**DSORG**=({**GS**|(*dsorglist*)})] [,**DEVD**=(*devlist*)]

DSORG=({**GS**|(*dsorglist*)})
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.

GS
specifies a data control block for graphics; this operand cannot be used in combination with any of the below.

dsorglist
You can specify one or more of the following values (each value 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. Although this option is supported, its use is not recommended.

IS
specifies a data control block for an indexed sequential data set. Although this option is supported, its use is not recommended.

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.

DEVD=(*devlist***)**
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.

devlist
You can specify one or more of the following values (each value must be separated by a comma). If you specify more than one value, they must have parentheses around them.

DA
Direct access device

PC
Online punch

PR
Online printer

RD
Online card reader or read punch feed

TA
Magnetic tape

MR
Magnetic character reader

ESETL—End Sequential Retrieval (QISAM)

Use of the ESETL macro is not recommended because it is a QISAM macro; we recommend you use VSAM instead.

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)
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 switches volumes automatically. You can specify volume positioning for magnetic tape with 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)
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 tape mark 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 your 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)
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)
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)
specifies the address of the area that contains the relative address (TTR) 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: 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 3 high-order bytes of register 15 are set to 0 and the low-order byte of register 15 contains the following return code:

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 three high-order bytes of registers 0 and 15 are set to 0, the low-order byte of register 15 contains one of the following return codes and the low-order byte of register 0 contains one of the following reason codes:

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)
specifies the address of the data control block for an opened data set to which the buffer pool has been assigned.

register—(2-12)
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)

Use of the FREEDBUF macro is not recommended.

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:

[*symbol*]	**FREEDBUF**	*decb address* ,{**K**|**D**} ,*dcb address*

decb address—RX-Type Address, (2-12), or (0)
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.

dcb address—RX-Type Address, (2-12), or (1)
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 releases an area of storage, previously acquired for a buffer pool for a specified data control block. The area must have been acquired either automatically (except when dynamic buffer control is used) or by executing a GETPOOL macro instruction. For queued access methods, you must issue a CLOSE macro instruction for all the data control blocks using the buffer pool *before* issuing the FREEPOOL macro instruction. For basic access methods, you can issue the FREEPOOL macro instruction when 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)
specifies the address of a data control block to which the buffer pool has been assigned.

GET—Obtain Next Logical Record (QISAM)

Use of the GET (QISAM) macro is not recommended; we recommend you use VSAM instead.

The GET macro instruction retrieves (reads) 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)
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)
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 *DFP: Customization*.
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 before 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 retrieves (reads) 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. You 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 your 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. You can use move mode 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 your work area. You cannot use the **TYPE=P** operand with data mode.

If the ISCII/ASCII translation routines are included when the operating system is generated, you can request translation by coding **LABEL=(,AL)** or **(,AUL)** in the DD statement, or 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 when the input buffer is full. For translation to occur correctly, all input data must be in ISCII/ASCII code.

The GET macro is written:

[*symbol*]	**GET**	{*dcb address*|*pdab address*} [,*area address*] [,**TYPE=P**]

dcb address—RX-Type Address, (2-12), or (1)
specifies the address of the data control block for the opened input data set being retrieved.

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

area address—RX-Type Address, (2-12), or (0)
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 nnnnn**K** 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 moves 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

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 209.

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. Use the FREEBUF macro instruction to return the buffer obtained to the buffer pool.

The GETBUF macro is written:

[*symbol*]	**GETBUF**	*dcb address* *,register*

dcb address—RX-Type Address, (2-12), or (1)
specifies the address of the data control block that contains the buffer pool control block address.

register—(2-12)
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 constructs a buffer pool 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)**}

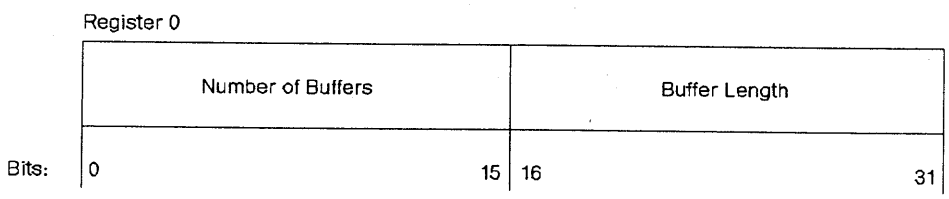
dcb address—RX-Type Address, (2-12), or (1)
specifies the address of the data control block to which the buffer pool is assigned. Only one buffer pool can be assigned to a data control block.

The value you specify may be either a positive or a negative value. If this operand has the high-order bit on (for example, to signify the last address in a list), this bit must be reset to zero. Otherwise, the address will be treated as a negative value.

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

buffer length—symbol, decimal digit, absexp, or (2-12)
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)
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 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.

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 is to be displayed in positions 2 through 7 of the display while the data set is open. The display is steady (not flashing) and is enclosed in parentheses. The display is also 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 is 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. For 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 are 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 is to be displayed in positions 2 through 7 while a data set is open. The display is steady (not flashing) and is enclosed in parentheses. The display is also 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 is 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. For 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 are 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 is to be displayed in positions 2 through 7 while a data set is open. The display is steady (not flashing) and is enclosed in parentheses. The display is also 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 is 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. For 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 is to 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 are 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. For return code = 08, 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 returns 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)
specifies the address of the data control block opened for the partitioned or sequential data set being processed.

TYPE={**ABS**|**REL**}
indicates whether the dcb address is a physical block identifier or a relative address.

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 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:

TTR is a 3-byte field indicating the relative track record address of the block positioned to.

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—If Type=ABS is Specified

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; if return code = 08, 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.

Completion Codes—If Type=REL is Specified

None.

OPEN—Logically Connect a Data Set (BDAM, BISAM, BPAM, BSAM, QISAM, and QSAM)

The OPEN macro instruction completes the specified data control block(s) and prepares for processing the data set(s) identified in the data control block(s). 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 *Programming Support for the IBM 3505 Card Reader and the IBM 3525 Card Punch.*

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)
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. You must code option 1 if also coding option 2. 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. You must specify **OUTPUT** or **OUTIN** when creating a data set.

Access Method	DEVICE TYPE: Magnetic Tape Option 1	Magnetic Tape Option 2	Direct Access Option 1	Direct Access Option 2	Other Types Option 1	Other Types 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]	
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.

Note: The **EXTEND**, **INOUT**, **OUTIN**, and **OUTINX** options are not allowed for ISO/ANSI/FIPS Version 3 tape processing.

Option 1	Meaning
EXTEND	The data set is treated as an **OUTPUT** data set, except that records are 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	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	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	The data set is treated as an **OUTIN** data set, except that records are 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.

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 you can specify 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 issuing 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 is 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 is 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** are 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 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 up to you, 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 can contain 31-bit addresses. Your program does not need to be executing in 31-bit addressing mode to use **MODE=31** in the OPEN macro. This parameter specifies the form of the parameter list, not the addressing mode of the program. The default, **MODE=24**, specifies a standard form parameter list with 24-bit addresses. **MODE=31** is not permitted if **TYPE=J** is specified. If **TYPE=J** is specified, you must use the standard form parameter list.

The standard form parameter list must reside below 16M, but the calling program may be above 16M. Assume that all ACBs and DCBs are below 16M.

The long form parameter list can reside above or below 16M. 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 is issued and the data set is not opened. The program is not 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.

The following errors in opening a DCB 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 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 (see note 1).

Notes:

1. You need to test bit 3 of the DCBOFLGS field in the data control block. Bit 3 is set to 1 if the data control block has been opened successfully, but 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 your 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.

2. Other errors detected by OPEN result in an ABEND being issued with a valid system completion code in the form x13 where x is a hex digit from 0 to F. See *System Codes* for the ABEND codes.

Return Codes from OPEN

When your program receives control after issuing an OPEN macro, the return code in register 15 indicates if all the data sets were opened successfully: The entries with errors are not restored, and cannot be reopened without restoration.

Return Code (15)	Meaning
0(0)	All data sets were opened successfully.
4(4)	All data sets were opened successfully, but one or more warning messages were issued.
8(8)	At least one data set (VSAM or non-VSAM) was not opened successfully; the access method control block was restored to the contents it had before OPEN was issued; or, if the data set was already open, the access method control block remains open and usable and is not changed.

Return Code (15)	Meaning
12(C)	A non-VSAM data set was not opened successfully when a non-VSAM and a VSAM data set were being opened at the same time; the non-VSAM data control block was not restored to the contents it had before OPEN was issued (and the data set cannot be opened without restoring the control block).

OPEN—List Form

The list form of the OPEN macro instruction constructs a data management parameter list. You can specify any number of operands (data control block addresses and associated options).

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:

[*symbol*]	**OPEN**	([*dcb address*],[(*options*)],...) **,MF=L** [**,TYPE=J**] [**,MODE=24|31**]

dcb address—A-Type Address

MF=L

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 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 up to you, 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 can contain 31-bit addresses. Your program does not need to be executing in 31-bit addressing mode to use **MODE=31** in the OPEN macro. This parameter specifies the form of the parameter list, not the addressing mode of the program. The default, **MODE=24**, specifies a standard form parameter list with 24-bit addresses. **MODE=31** is not permitted if **TYPE=J** is specified. If **TYPE=J** is specified, you must use the standard form parameter list.

The standard form parameter list must reside below 16M, but the calling program may be above 16M. Assume that all ACBs and DCBs are below 16M.

The long form parameter list can reside above or below 16M. 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 is issued and the data set is not opened. The program is not 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***)** [**,TYPE=J**] [**,MODE=24|31**]

dcb address—RX-Type Address or (2-12)

MF=(E,*data management list address***)**

The **MF=E** 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

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 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 up to you, 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 can contain 31-bit addresses. Your program does not need to be executing in 31-bit addressing mode to use **MODE=31** in the OPEN macro. This parameter specifies the form of the parameter list, not the addressing mode of the program. The default, **MODE=24**, specifies a standard form parameter list with 24-bit addresses. **MODE=31** is not permitted if **TYPE=J** is

specified. If **TYPE=J** is specified, you must use the standard form parameter list.

The standard form parameter list must reside below 16M, but the calling program may be above 16M. Assume that all ACBs and DCBs are below 16M.

The long form parameter list can reside above or below 16M. 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 is issued and the data set is not opened. The program is not 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.

The data sets with errors are not restored, and cannot be reopened without restoration.

PDAB—Construct a Parallel Data Access Block (QSAM)

The PDAB macro instruction is used 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**=*absexp*

MAXDCB=*absexp* (maximum value is 32767 bytes)
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 generates 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 245.

The name of the dummy control section generated by a PDABD macro instruction is IHAPDAB. A USING instruction specifying IHAPDAB and a dummy section base register containing the address of the actual parallel data access block should come before any of the symbolic names provided by the dummy section. You should use the PDABD macro instruction once within any assembled module; however, you can use the resulting symbolic names for any number of parallel data access blocks by changing the address in the dummy section base register. You can code the PDABD macro instruction 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 the POINT macro, test for completion all input and output operations using the same data control block. 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)
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)
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, you can use the POINT macro instruction 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 fail. (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; if return code = 08, 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 controls 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 tests for an overflow condition on the specified channel (either channel 9 or channel 12) of the printer carriage control, and either skips the printer carriage to the line corresponding to channel 1, or transfers 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. You should issue the PRTOV macro 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), issue the PRTOV macro instruction before the first PUT or WRITE macro instruction only. You should issue the PRTOV macro instruction only when the device type is an online printer. You cannot use PRTOV 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)
specifies the address of the data control block opened for output to an online printer or 3525 card punch with a print feature.

9
12
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 *Programming Support for the IBM 3505 Card Reader and the IBM 3525 Card Punch.*

overflow exit address—RX-Type Address or (2-12)
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)

Use of the PUT (QISAM) macro is not recommended; we recommend you use VSAM instead.

The PUT macro instruction writes 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. Using 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:

[*symbol*]	**PUT**	*dcb address* [,*area address*]

dcb address—RX-Type Address, (2-12), or (1)
specifies the address of the data control block for the opened ISAM data set.

area address—RX-Type Address, (2-12), or (0)
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 cannot 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 209.

PUT—Write Next Logical Record (QSAM)

The PUT macro instruction writes a record in a sequential data set. Various modes are available and are specified in the DCB macro instruction. The modes are locate mode, move mode, and data mode. In the locate mode, the address of an area within an output buffer is returned in register 1 after the PUT macro instruction is executed. You should then construct, at this address, the next sequential record or record segment. If the move mode is used, the PUT macro instruction moves a logical record 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 is 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, you can request translation by coding **LABEL=(,AL)** or **(,AUL)** in the DD statement, or 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:

[*symbol*]	**PUT**	*dcb address* [,*area address*]

dcb address—RX-Type Address, (2-12), or (1)
specifies the address of the data control block for the data set opened for output.

area address—RX-Type Address, (2-12), or (0)
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 you specify locate mode, omit the *area address* operand. 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 is returned in register 0. In this case, you must segment the records and process them in 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. Then, move 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 returns the address of the next buffer in register 1. After this address is given to you, QSAM always will count this address as a valid record. You 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 is written to the data set. After issuing an FEOV macro (for multivolume data sets), you must reinitialize register 1 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. You must place the total data length in the DCBPRECL (not the DCBLRECL) field of the data control block before issuing the PUT macro instruction.

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 *DFP: Customization*.

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 your user 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 record descriptor word in variable-length records must not be changed.

The PUTX macro is written:

[*symbol*]	**PUTX**	*dcb address* [,*input dcb address*]

dcb address—RX-Type Address, (2-12), or (1)
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)
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 *DFP: Customization*.

READ—Read a Block (BDAM)

Use of the READ (BDAM) macro is not recommended; we recommend you use a device-independent access method such as BSAM, BPAM, or QSAM instead.

The READ macro instruction retrieves a block from a data set and places it 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 209, 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* ,{**DI**[**F**|**X**][**R**|**RU**]} {**DK**[**F**|**X**][**R**|**RU**]} ,*dcb address* ,{*area address*|**'S'**} ,{*length*|**'S'**} ,{*key address*|**'S'**|**0**} ,*block address* [,*next address*]

decb name—symbol
specifies the name assigned to the data event control block created as part of the macro expansion.

{**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**, **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)
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'**
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'**
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. 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 (DECB) before the READ is issued.

key address—A-Type Address, (2-12), **'S'**, or **0**
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*, You can also code **'S'** 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)
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 as an actual device address, and the field must be 8 bytes long.

next address—A-Type Address or (2-12)
specifies the address of the storage area in which the system places the relative address of the next block. 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)

Use of the READ (BISAM) macro is not recommended; we recommend you use VSAM instead.

The READ macro instruction retrieves an unblocked record, or a block containing a specified logical record, 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 209.

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* ,{**K**|**KU**} ,*dcb address* ,{*area address*|**'S'**} ,{*length*|**'S'**} ,*key address*

decb name—symbol
specifies the name assigned to the data event control block (DECB) created as part of the macro expansion.

{**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)
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'**
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 infor-

mation 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*.

length—symbol, decimal digit, absexp, (2-12), or **'S'**
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'** *must* be coded.

key address—A-Type Address or (2-12)
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 retrieves a block from a data set and places it in a designated area of storage (input area). 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 209, 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* ,{**SF**|**SB**} ,*dcb address* ,*area address* [,*length*|,**'S'**]

decb name—symbol
specifies the name assigned to the data event control block (DECB) created as part of the macro expansion.

{**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)
specifies the address of the data control block for the opened data set to be read.

area address—A-Type Address or (2-12)
specifies the address of the problem program area in which the block 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 block 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'**
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 block 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)

Use of the READ (BDAM) macro is not recommended; we recommend you use a device-independent access method such as BSAM, BPAM, or QSAM instead.

The READ macro instruction retrieves a block from a data set and places it in a designated area of storage. The data set is a BDAM data set and its record format is unblocked variable-length spanned records. You must specify **BFTEK=R** 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 209, 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* **,SF** *,dcb address* *,area address*

decb name—symbol
specifies the name assigned to the data event control block (DECB) created as part of the macro expansion.

SF
specifies normal, sequential, forward retrieval.

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

area address—A-Type Address or (2-12)
specifies the address of the area in which the block 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 as 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 209.

The description of the standard form of the READ macro instruction explains 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

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 explains 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
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)

Use of the RELEX macro is not recommended because it uses the device-dependent access method BDAM. We recommend you use a device-independent access method such as BSAM, BPAM, or QSAM instead.

The RELEX macro instruction releases 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: You can also use a WRITE macro instruction that specifies either **DIX** or **DKX** 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)
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)
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 immediately releases 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 following 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)
specifies the address of the data control block for the opened input data set.

SETL—Set Lower Limit of Sequential Retrieval (QISAM Input)

Use of the SETL macro is not recommended because it is a QISAM macro; we recommend you use VSAM instead.

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. You must issue an ESETL macro instruction 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)
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 non-deleted 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.

lower limit address—RX-Type Address, (2-12), or (0)
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, the addressed 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 the addressed area.

SETL Exit

The error analysis (SYNAD) routine is given control if the operation could not be completed successfully. For information on how the exception condition code and general registers are set, see *DFP: Customization*. If the SETL macro instruction is not reissued, retrieval starts at the beginning of the data set.

SETPRT—Printer Setup (BSAM, QSAM, and EXCP)

3800 Printers and SYSOUT Data Sets

The SETPRT macro instruction is used to initially set or dynamically change the printer control information for the IBM 3800 Printing Subsystem and SYSOUT data sets. You can use SETPRT with any 3800 model printer that is allocated to the program (not to JES). You can also use SETPRT when creating SYSOUT data sets. You may change the following control information 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).

Besides changing the control information, you can do the following:

- Supply your own 3800 load modules in a PDS 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 directly allocated printer (**PRTMSG** parameter).
- Control the scheduling of SYSOUT segment printing (**DISP** parameter).

To use all-points addressability when operating the 3800 Model 3, 6, or 8, PSF libraries (for example, SYS1.FONTLIB, SYS1.FDEFLIB, SYS1.PDEFLIB) are used instead of SYS1.IMAGELIB.

For additional information on how to use the SETPRT macro instruction with the IBM 3800 Model 3, 6, or 8, see *IBM 3800 Printing Subsystem Models 3 and 8 Programmer's Guide*.

Non-3800 Printers

For printers other than the IBM 3800 Printing Subsystem, SETPRT controls 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)

- Bypassing automatic forms positioning.

The SETPRT macro automatically positions forms in the printer to the first line of a new page when a new FCB is requested. If you wish to position the form yourself, specify the **N** option of the **FCB** parameter and insert the new form, matching the top of its page to the same position as the old form occupied.

This is how the SETPRT macro aligns a new form: If the FCB is different from the one currently in the printer, the old FCB and the current position within it is read from the printer. If the old form is not already at the top of a page, a temporary FCB is constructed and loaded back into the printer. A skip to 1 command is then executed to move the old form to the top of a new page. The requested FCB is then loaded into the printer. SETPRT's preparation is now complete. The new FCB and the old form are now at the first line of a new page. Printing is ready to proceed. If you wish to bypass automatic forms positioning, use the **N** option of the **FCB** Parameter.

4248 Printers

For the 4248 printer, the SETPRT macro instruction is also used to change the following control information:

- Activation, deactivation, and positioning of horizontal copy (**COPYP** parameter)
- Speed of the printer (**PSPEED** parameter).

All Printers

When BSAM is used, all write operations must be checked for completion before the SETPRT macro instruction is issued. Otherwise, an incomplete write operation may be purged.

Issuing the SETPRT macro instruction for a device other than a SYSOUT data set, or a UCS printer, or the IBM 3800 Printing Subsystem results in an error return code.

Note: A permanent error on a SETPRT macro causes one or both of the first two bits of the DCBIFLGS field to be set on. A cancel key or a paper jam that requires a printer subsystem-restart sets in the DCBIFLGS field the lost data-indicator bit, DCBIFLDT. Before reissuing a SETPRT macro, you must reset these bits to zero.

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*] [,**COPYP**={*position*|**0**}] [,**DISP**={**SCHEDULE**|**NOSCHEDULE**|**EXTERNAL**}] [,**FCB**={*imageid*|**A**(*address*)|**R**(*register*)} ({*imageid*|**A**(*address*)|**R**(*register*)}[,{**V**|**A**}[,**N**]]) [,**FLASH**={**NONE**|*name*} {**NONE**|([*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**}] [,**PSPEED**={**L**|**M**|**H**|**N**}] [,**REXMIT**={**N**|**Y**}] [,**UCS**={*csc*} {(*csc*,{**F**|**F**,**V**|**V**})}]

dcbaddr—A-Type Address or (2-12)
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**}
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 prints a message at the system console telling the operator to thread the paper again if needed. This operand is effective for all IBM 3800 printers only.

CHARS={*name*\|**A**(*address*)\|**R**(*register*)}
{({*name*\|**A**(*address*)\|**R**(*register*)},...)}
specifies one to four character arrangement tables to be used when printing a data set. This operand is effective for all IBM 3800 models printers.

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 effective for the IBM 3800 printer only.

COPYNR=*number*
specifies the starting copy number for this transmission. *number* is a value from 1 to 255. This operand defaults to a value of 1 if not specified. This operand is effective for an online IBM 3800 printer only.

COPYP={*position*|**0**}
activates or deactivates the horizontal copy feature of the 4248 printer. This overrides the horizontal copy offset in the specified FCB. (If no FCB is specified, the horizontal copy offset in the already loaded FCB is overridden.) **COPYP** also controls horizontal copy capabilities with 3211 FCBs that are loaded in a 4248 printer.

position
is a decimal number from 2 to 168 indicating the print position where the horizontal copy starts. If your 4248 printer has only 132 print positions installed, the maximum number you should specify here is 132. When horizontal copy is activated, the maximum amount of data that can be sent to the printer is equal to the size of the smaller of the two copy areas. If the two copy areas are equal, the maximum amount of data that can be sent is equal to half the number of print positions.

For example, if you specify **COPYP=101** for a 4248 printer with 132 print positions, the maximum amount of data that can be sent to the printer is 32 bytes. (Thirty-two bytes is equal to the smaller copy area, from position 101 to position 132.) If you specify **COPYP=67** for a 4248 printer with 132 print positions, the maximum amount of data that can be printed is 66 bytes. (Sixty-six bytes is equal to half the number of print positions.)

If **COPYP=***position* is specified and a 3211 format FCB is being used, the 3211 format FCB is converted to 4248 format FCB and the specified offset value is inserted.

Note: **COPYP=***position* is not available with the IBM 3262 Model 5 printer.

0
specifies that no horizontal copy is to be made. Any offset value in the specified or already loaded FCB is be overridden.

Note: Channel programs that are used when horizontal copy is activated *must* have the suppress length indication (SLI) bit set. For information on the SLI bit, see *IBM System/370 XA Principles of Operation*.

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 abbre-

viate the parameters to **S**, **N**, and **E**, respectively. This operand is effective for the IBM 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**}[,**N**]**)**
specifies that the forms control buffer (FCB) is to be selected from the image library. 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. Note that the 4248 accepts both 3211 and 4248 format FCBs. For descriptions of the standard forms control images for the 3203 and 3211, 3262 Model 5 or 4245, see *System—Data Administration*. For a description of the 4248 FCB, see *Utilities*. For more information about 3800 FCB modules, see *Utilities*.

A(*address***)**
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*.)

Note: This subparameter is effective for online IBM 3800 Model 1 printers.

R(*register***)**
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*.)

Note: This subparameter is effective for online IBM 3800 Model 1 printers.

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 is ignored if specified for the IBM 3800 printer.

N

bypasses automatic forms positioning. This subparameter is ignored if specified for the IBM 3800 printer. **N** is not available via JCL and, thus, cannot be used when opening an online because all SETPRT parameters are obtained from the JCL at open.

FLASH={**NONE**|*name*}
{**NONE**|([*name*],*count*)}

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 lets you specify the number of copies on which the overlay is to be printed (flashed). If you omit this operand for a directly attached printer, flashing stops. If you omit this operand when doing a SETPRT while generating SYSOUT data, the **FLASH** parameters previously in effect for this data set are used. This operand is effective for the IBM 3800 printer only.

NONE

is valid only when using SETPRT while generating SYSOUT data, and causes zero copies to be flashed. If flashing is resumed in a later SETPRT, a message is generated by JES regarding the insertion of the forms overlay frame, even if no change in the forms overlay frame is necessary.

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 is to be printed, beginning with the first copy. The number of copies printed is not greater than the number of copies specified by the **COPIES** operand.

For a directly attached printer: No copies are flashed if you specify a flash count of zero. If you specify a nonzero flash count and omit the name of the forms overlay frame, the operator is not requested to insert a frame. Whatever frame is inserted is used.

During the generation of SYSOUT data: If you specify a flash count of zero, the flash count previously in effect for the data set is used. If you specify a nonzero flash count and omit the name of the forms overlay frame, the operator is not requested to insert a frame except when flashing has stopped. If flashing has stopped, a message from JES requests the operator to insert a new frame. Then the flashing of the forms will resume using the count specified in the flash count parameter.

INIT={**N**|**Y**}

When **INIT=Y** is specified for an online IBM 3800 printer, it initializes the control information in the printer with a folded character arrangement table: the 10-pitch Gothic character set (12 pitch for the IBM 3800 Models 3,6, and 8), and a six lines per inch FCB corresponding to the forms size in the printer. **COPIES** and **COPYNR** are initialized to 1, **FLASH** and **MODIFY** are cleared, and **BURST** is initialized to **N** (continuous forms).

When **INIT=Y** is specified for a SYSOUT data set, other parameters not specified on the same invocation are reset, meaning the JES default will be used. For **INIT=N**, all control information for the IBM 3800 printer remains

unchanged. Any parameters included on the same macro statement as the **INIT** operand are processed after printer initialization has been completed. This operand is effective for the IBM 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.

Note: This operand is effective for online IBM 3800 Model 1 printers.

MODIFY={*name*|**A(***address***)**|**R(***register***)**}
{({*name*|**A(***address***)**|**R(***register***)**},*trc***)**}
identifies the copy modification module and an associated character arrangement table module to be used when modifying the data to be printed.

Note: This operand is effective for online IBM 3800 Model 1 printers.

name
is the 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***)**
specifies an in-storage address of the user-supplied copy modification module. For information on the format of the module, see *Utilities*. This subparameter is effective for the IBM 3800 Model 1 printer.

R(*register***)**
specifies the register that contains an in-storage address of the user-provided copy modification module. For information on the format of the module, see *Utilities*. This subparameter is valid for the IBM 3800 Model 1 printer.

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 effective with the IBM 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})}
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.

FOLD or **F**
specifies that printing is in fold mode. This subparameter is ignored if specified for the IBM 1403 or IBM 3800 printer. For 1403 fold mode, use **FOLD** option under the **UCS** operand.

UNFOLD or **U**
specifies that printing is in normal mode; this operand causes fold mode to revert to normal mode. This subparameter is ignored if specified for the IBM 1403 or IBM 3800 printer. Because **UCS** processing occurs after **OPTCD** processing, if **FOLD** is specified in the **UCS** operand, fold mode is set. If **FOLD** is not coded, unfold is set.

PRTMSG={N|Y}
allows printer error messages to be printed for the programmer on the IBM 3800. This operand is effective with IBM 3800 only.

N
specifies not to print error messages on the IBM 3800.

Y
specifies to print error messages on the IBM 3800. **Y** is the default.

PSPEED={L|M|H|N}
is effective for the 4248 printer only, and is ignored for all other printers. The **PSPEED** operand specifies the printer's speed, which affects print quality. **LOW** speed produces the best quality. The **PSPEED** operand is used to set the printer's speed or override that set in the FCB. If no FCB is specified, the **PSPEED**, if any, in the already loaded FCB is used.

L or **LOW**
sets the printer speed to 2200 lines per minute.

M or **MEDIUM**
sets the printer speed to 3000 lines per minute.

H or **HIGH**
sets the printer speed to 3600 lines per minute.

N or **NOCHANGE**
indicates that the speed at which the printer is currently running is to remain the same no matter what is specified in the requested FCB, or if none is specified, in the already loaded FCB.

Actual printer speed may vary. For information on determining the exact printer speed, see *IBM 4248 Printer Model 1 Description.*

REXMIT={**N**|**Y**}
specify **REXMIT=Y** to modify 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 ignores all other parameters in the parameter list.

UCS={*csc*}
{(*csc*,{**F**|**F**,**V**|**V**})}
specifies the character set image to be used. This operand is ignored if specified for the IBM 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**

IBM 4245 Printer: **AN21**, **AN31**, **HN21**, **HN31**, **PL21**, **PL31**, **GN21**, **RN21**, **RN31**, **TN21**, **SN21**, **FC21**, **KA21**, and **KA22**

4248 Printer: **40E1**, **40E2**, **4101**, **4102**, **4121**, **4122**, **41C1**, **41C2**, **4181**, **4201**, **4061**, **40C1**, **4161**, **4041**, and **4042**

Note: There are no standard IBM character sets supplied for the IBM 3262 Model 5 printer.

The 4245 and 4248 printers load their own images on recognition of the mounted band. The image table provides a correspondence between the band identification and the character set code.

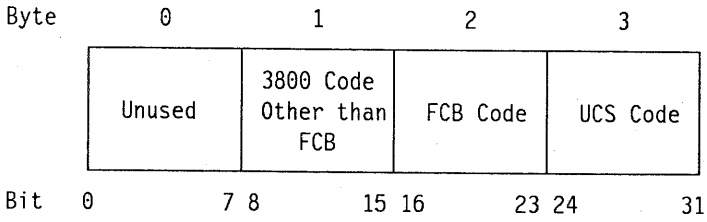
For a description of the 4245 and 4248 UCS image tables and information on adding user-defined entries to an image table, see *System—Data Administration*.

FOLD or **F**
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. Unless **FOLD** is specified, **UNFOLD** mode is set.

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.



The return codes in the figures that follow are in hexadecimal.

- Return codes 0 through 24 apply to all printers.
- Return codes 28 through 4C apply to the 3800 printer only. There is one exception; return code 48 also applies to the IBM 3262 Model 5 and the IBM 4248 printer.
- Return code 50 applies to SYSOUT data sets only for any printer.

Return Codes 0 to 14

Figure 1 shows the hexadecimal return codes 00 through 14 for specific printers.

Figure 1 (Page 1 of 3). SETPRT Return Codes 00 to 14

3800 Code Other than FCB (Byte 1)	FCB Code (Byte 2)	UCS Code (Byte 3)	Meaning
00	00	00	Successful completion.
00	00	04	The operator canceled the UCS request for one of the following reasons: • The UCS image could not be found in SYS1.IMAGELIB. • The requested train or band was not available.
00	04	00	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 your 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.

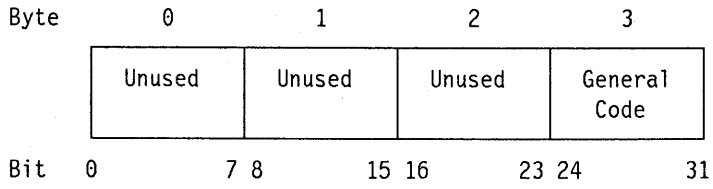
Figure 1 (Page 2 of 3). SETPRT Return Codes 00 to 14

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 170.
00	00	08	A permanent I/O error was detected when the BLDL macro instruction was issued to locate a UCS image or image table in SYS1.IMAGELIB.
00	08	00	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 170.
00	00	0C	A permanent I/O error was detected while loading the printer's UCS buffer, or displaying a message on the 4248 printer.
00	0C	00	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 7 on page 172.

Figure 1 (Page 3 of 3). SETPRT Return Codes 00 to 14

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 170.
00	00	10	A permanent I/O error was detected during UCS verification display or while reading the UCS buffer.
00	10	00	A permanent I/O error was detected during FCB verification display.
00	00	14	The operator canceled the UCS request because an improper character set image was displayed for visual verification.
00	14	00	The operator canceled the FCB request because an improper forms control image was displayed for visual verification.

The illustration below shows how the four bytes of register 15 are used for all printers.



Return Codes 18 to 50

Figure 2 shows the hexadecimal return codes 18 through 50 for all printers.

Figure 2 (Page 1 of 2). SETPRT Return Codes 18 to 50

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 4245, a possible lost data condition was detected. For a 3800, message IEC173I indicates which of the above errors has occurred. Register 0 contains a reason code identifying whether data was lost. For an explanation, see Figure 4 on page 171.
20	Not enough storage was available for opening the SYS1.IMAGELIB, or, for a 3800 printer, not enough storage was available to contain the control blocks for a user library, or insufficient storage was available for SETPRT.
24	SYS1.IMAGELIB (or, for the 3800 printer, a user library) cannot be opened to load the specified module. Either a permanent I/O error occurred or the SYS1.IMAGELIB was mounted or cataloged incorrectly.
28	The operator canceled the forms overlay request.
2C	The operator canceled the paper threading request.
30	More writable character generation modules (WCGMs) were 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.

Figure 2 (Page 2 of 2). SETPRT Return Codes 18 to 50

Return Code (Byte 3)	Meaning
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.
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. For a 4248, a possible lost data condition was detected. Register 0 contains a reason code identifying which of the above conditions occurred. See Figure 5 on page 171 for an explanation.
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 170.
50	When a SETPRT was issued to a Direct Attach (an online 3800 Model 3, 6 or 8 printer) or a SYSOUT data set, there was a failure in one of the following: • 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. For an explanation of the reason codes associated with return code 50, see Figure 6 on page 171.

Reason Codes

All 3800 Printers

The following illustration shows the contents of register 0, which includes the GCM ID, the CAT ID, and the reason code.

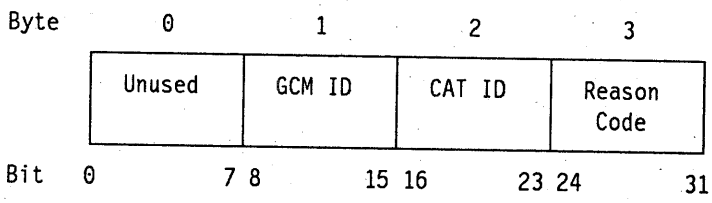


Figure 3 shows the hexadecimal reason codes for the IBM 3800 Model 1 and the other 3800 models in compatibility mode. These reason codes, returned in register 0, are in addition to return codes 04, 08, 0C, and 4C returned in register 15.

Figure 3. Reason Codes for IBM 3800 Printers (with Return Codes 04, 08, 0C, 4C)

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.

3800 Printers and the 4245 Printer

These reason codes apply to all 3800 printers and the IBM 4245 printer. Figure 4 shows the reason codes besides return code 1C returned in register 15. The reason code is placed in byte 3 of register 0.

Figure 4. Reason Codes for All Printers (for Return Code 1C)

Reason Code (Byte 3)	Meaning
00	Indicates no data lost.
04	Indicates data has been lost.

Figure 5 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.

Figure 5. Reason Codes for 3800 Printers and 4248 Printer (for Return Code 48)

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 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.

Figure 6 (Page 1 of 2). Reason Codes for Return Code 50

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 load module IDs in SYS1.IMAGELIB are allowed for SYSOUT setup.
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.

Figure 6 (Page 2 of 2). Reason Codes for Return Code 50	
Reason Code (Byte 3)	**Meaning**
18	An ENQ macro failed. The ENQ was issued by SETPRT processing.
1C	More than one DCB is open for the SYSOUT data set.

All Non-3800 Printers

Figure 7 shows the reason code in addition to completion code 0C00.

Figure 7. Reason Codes for Non-3800 Printers (for Completion Code 0C00)	
Reason Code (Byte 3)	**Meaning**
00	The I/O error was not caused by a load check.
04	FCB load failed because of a load check. Probably caused by invalid FCB contents.

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*] [,**COPYP**=**{***position*|**0}**] [,**DISP**=**{SCHEDULE|NOSCHEDULE|EXTERNAL}**] [,**FCB**=**{***imageid***}** **(***imageid*,**{V|A}**[,**N**]) [,**FLASH**=**{NONE|***name***}** **{NONE|([***name***],***count***)}**] [,**INIT**=**{N|Y}**] [,**LIBDCB**=*dcbaddress*] [,**MODIFY**=**{***name***}** **{(***name*,*trc***)}**] [,**MSGAREA**=*address*] [,**OPTCD**=**{B|U}** **{({B|U},{F|U})}**] [,**PRTMSG**=**{N|Y}**] [,**PSPEED**=**{L|M|H|N}**] [,**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.

COPYP={*position*|**0**}
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**}[,**N**])
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={**NONE**|*name*}
{**NONE**|([*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.

PSPEED={**L**|**M**|**H**|**N**}
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*|*}] [,**COPYP**={*position*|**0**}] [,**DISP**={**SCHEDULE**|**NOSCHEDULE**|**EXTERNAL**}] [,**FCB**={*imageid*|**A**(*address*)|**R**(*register*)} ({*imageid*|**A**(*address*)|**R**(*register*)[,{**V**|**A**}[,**N**]]) {*}] [,**FLASH**={**NONE**|*name*} {([**NONE**|*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**}] [,**PSPEED**={**L**|**M**|**H**|**N**}] [,**REXMIT**={**N**|**Y**|*}] [,**UCS**={*csc*} {(*csc*,{**F**|**F**,**V**|**V**})}] ,**MF**=(**E**,*data management list address*)

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 effective for the IBM 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.

COPYP={*position*|**0**}
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*|**A(***address***)**|**R(***register***)**}
({*imageid*|**A(***address***)**|**R(***register***)**}[,{**V**|**A**}[,**N**])
{*}
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={**NONE**|*name*}
{**NONE**|([*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**) are 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.

PSPEED={**L**|**M**|**H**|**N**}
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*)
specifies that the execute form of the SETPRT macro instruction is used, and an existing data management parameter list is used.

E

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

STOW—Update Partitioned Data Set Directory (BPAM)

The STOW macro instruction updates 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 is a member name, the system writes an end-of-data indication following the member. If the entry to be replaced is open for update, the system rewrites the directory entry. All input/output operations using the same data control block must have previously been tested for completion.

You can use the STOW macro instruction only when the data set is opened for OUTPUT, UPDAT or OUTIN (BSAM). It is best to use a BPAM DCB with the STOW macro. See *Data Administration Guide*, "Processing a Partitioned Data Set" for more information on using the STOW macros with different types of DCBs.

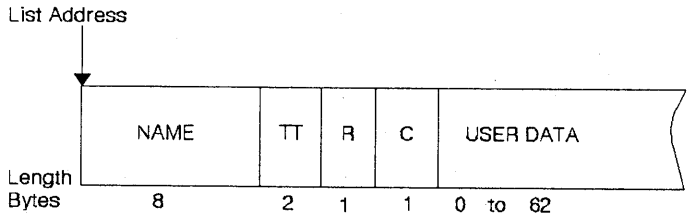
The STOW macro is written:

[*symbol*]	**STOW**	*dcb address* *,list address* [*,directory action*]

dcb address—RX-Type Address, (2-12), or (1)
specifies the address of the data control block for the opened partitioned data set.

list address—RX-Type Address, (2-12), or (0)
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 fields 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 fields when a member name is being added or replaced. Issue the FIND macro to locate the member before using STOW to replace it.

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 your data field.
3-7	Indicate the total number of halfwords in the user data field.

USER DATA FIELD: The user data field contains the user data from the directory entry. You can use the user data field to provide variable data as input to the STOW macro.

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.

directory action—[**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 two low-order bytes. The high-order bytes of both registers are set to 0.

The following is a list of return codes contained in register 15:

Return Codes (15)	Directory Action A	C	D	R
00 (X'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 (X'04')	The directory already contains the specified name.	The directory already contains the specified new name.	—	—
08 (X'08')	—	The specified name could not be found.	The specified name could not be found.	The specified old name could not be found.
12 (X'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.
16 (X'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.
20 (X'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.
24 (X'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 attempting to write the 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 *System—Data Administration* and *DFP: Customization*)

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 later PUT or WRITE macro instruction. The message consists of EBCDIC information and is in 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, you should code the SYNADAF macro at the beginning of the routine. (See *DFP: 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, you must use a SYNADRLS macro instruction 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**=*dcbaddr*]} {**EXCP** [,**PARM1**=*iobaddr*]} {**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.

Note: BDAM, BISAM, EXCP, and QISAM are not recommended.

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, dcbaddr*, **or** *iobaddr*—(2-12), (0), or RX-Type
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 for **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
```

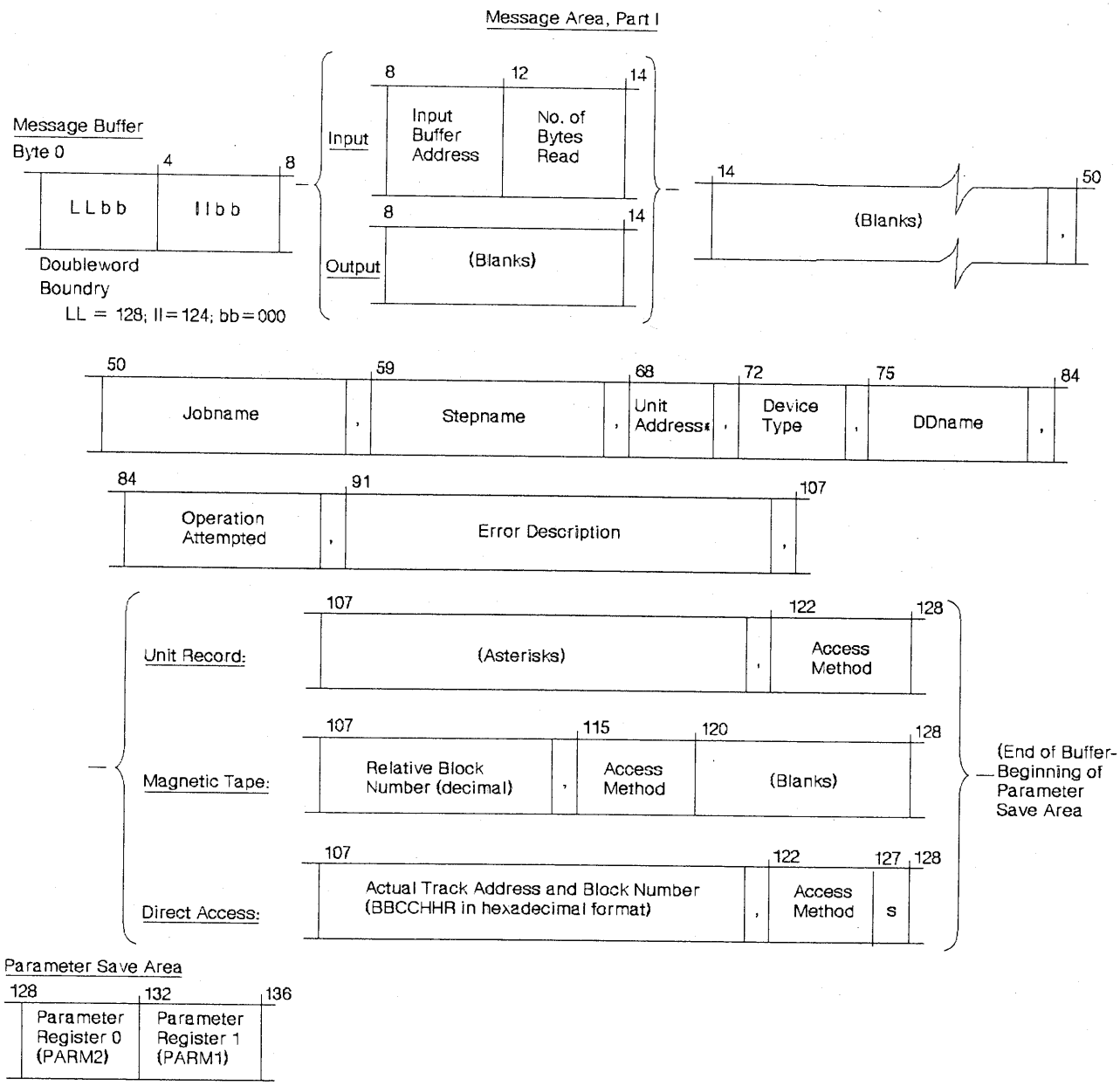
Completion Codes

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. If no data was transmitted, or if the access method is QISAM, bytes 8 through 13 contain blanks or binary zeros.
2. The unit address field (bytes 68 through 70) contains the letters 'JES' if the data set is SYSIN or SYSOUT.
3. 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.

4. 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.
5. If the access method is BISAM, bytes 68 through 70, 84 through 89, and 107 through 120 contain asterisks.
6. If the access method is BDAM, and if the error was an invalid request, bytes 107 through 120 contain EBCDIC zeros.

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

Tape Data Sets

The SYNCDEV macro instruction allows you to synchronize data to 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:

- 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:

[*symbol*]	**SYNCDEV**	**DCB**=*addr* [,{**ABUFBLK**=*addr*| **BUFBLK**={*maximum buffer depth*|*0*}}] [,**INQ**={**YES**|**NO**}]

The following describes the operands that can be specified for 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.

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 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.

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.

MF=(**E**,*addr*)
specifies the execute form of SYNCDEV.

addr—A-Type address, RX-Type address, or (2-12)
specifies the 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')	System error occurred.
04 (X'04')	04 (X'04')	Possible system error.
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')	00 (X'00')	Permanent I/O error during read block ID or synchronize command.
12 (X'0C')	00 (X'00')	Permanent I/O error on the last channel program with loss of data (for tape data only). **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 nnnnn**K** 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)
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 209. 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 *number of events* is 0 when the last event control block is checked, control is returned to the instruction following the WAIT macro instruction.
- If *number of events* is not 0 when 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|NO]
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 on expiration of the time limits that apply to it.

Access method ECBs are maintained entirely by the access methods and supporting control program facilities. You may inspect access method ECBs, but should never modify them.

WRITE—Write a Block (BDAM)

Use of the WRITE (BDAM) macro is not recommended; we recommend you use a device-independent access method such as BSAM, BPAM, or QSAM instead.

The WRITE macro instruction adds or replaces 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 to the problem program 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 209, 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* ,{**DA**[**F**]} {**DI**[**F**|**X**]} {**DK**[**F**|**X**]} ,*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.

{**DA**[**F**]}
{**DI**[**F**|**X**]}
{**DK**[**F**|**X**]}
The *type* operand 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 block is to be added to the data set. The search for available space starts on the track indicated by 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*. For a description of adding records with extended search, see the **LIMCT** operand of the DCB macro.)

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 209.

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 of the DCB 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 is 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. Specify **'S'** instead of an address only if the key is contained in an area acquired by dynamic buffering. If the key is not to be written or used as a search argument, specify zero 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 opera-

tion. 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)

Use of the WRITE (BISAM) macro is not recommended; we recommend you use VSAM instead.

The WRITE macro instruction adds or replaces 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 209, 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* ,{**K**|**KN**} ,*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.

{**K**|**KN**}
The *type* operand 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.

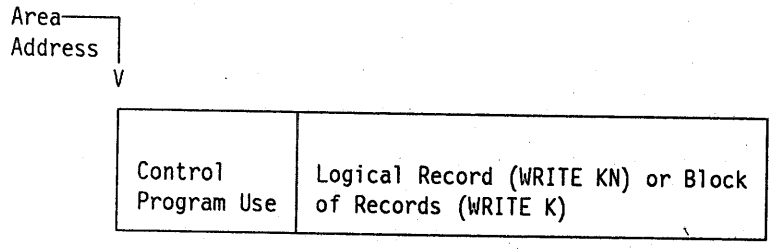
KN
specifies that a new record is to be written, or a variable-length record is to be rewritten with a different length. All records or blocks of records read using READ **KU** macro instructions for the same data control block must be written back before a new record can be added, except when the READ **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. The addressed 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. The addressed 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 is to 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 adds or replaces 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 209, 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 translates the output data in the output buffer 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. You can code **'S'** 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. Omit the *length* operand 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 Direct Data Set with BSAM)

Use of the WRITE (BDAM) macro is not recommended; we recommend you WRITE (BSAM) or WRITE (QSAM) instead.

The WRITE macro instruction adds 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 to the problem program 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 209, 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* ,{**SF**|**SFR**|**SD**|**SZ**} ,*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.

{**SF**|**SFR**|**SD**|**SZ**}
The *type* operand 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. You must specify **DSORG=PS** (or **PSU**) and **MACRF=WL** 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 Direct Data Set with BSAM)

After the write has been scheduled and control has been returned to your problem 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 (15)	Meaning		
	Fixed-Length	*Variable or Undefined-length*	*Variable or Undefined-length*
	(**SF** or **SD**)	(**SF** or **SFR**)	(**SZ**)
00 (X'00')	Block is 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 is 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 (X'04')	Block is 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 (X'08')	Block is 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 macro instruction is issued on a one-track secondary extent.
12 (X'0C')	Block is not written; issue a CHECK macro instruction for the previous WRITE macro instruction, then reissue the WRITE macro instruction.	Block is not written; issue a CHECK macro instruction for the previous WRITE macro instruction, then reissue the WRITE macro instruction.	Block is not 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 on 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 as 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 209.

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
specifies that the WRITE macro instruction is used to create a data event control block that is to 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
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.

See "Tables for Translating Codes" on page 226 for the ISCII/ASCII to EBCDIC and EBCDIC to ISCII/ASCII translation codes. When translating EBCDIC code to ISCII/ASCII code, all EBCDIC code not having an ISCII/ASCII equivalent is translated to X'1A'. When translating ISCII/ASCII code to EBCDIC code, all ISCII/ASCII code not having an EBCDIC equivalent is translated to X'3F'. 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 (DCB), or in the data event control block (DECB) (basic access methods). 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 you code a DCBD macro instruction, the exception code in a data control block can be addressed as two 1-byte fields, DCBEXCD1 and DCBEXCD2. For more information, see *DFP: 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 *DFP: 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
NOTE			X	X		
OPEN	X	X	X	X	X	X
PDAB						X

Macro Instruction	BDAM	BISAM	BPAM	BSAM	QISAM	QSAM
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
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. You can use these values to determine the maximum block size and logical record length for a given device, and 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
4245 Printer	132 bytes
4248 Printer	132 or 168 bytes
3262 Model 5 Printer	132 bytes

Magnetic Tape Units

Tape	Capacity
3410 Magnetic Tape Units (7 track and 9 track)	32760 bytes
3420 Magnetic Tape Units (7 track and 9 track)	32760 bytes
3430 Magnetic Tape Units	32760 bytes
3480 Magnetic Tape Subsystem	32760 bytes

Direct Access Devices

The following figure lists the physical characteristics of direct access storage devices.

Figure 8. DASD Physical Characteristics

DASD Characteristics			Block Size			
			512	1024	2048	4096
Type	Trk/Cyl	Cyl/Vol	Number of Physical Blocks per Track[2]			
2305-2	8	96	20	12	6	3
3330-1	19	404	20	11	6	3
3330-11	19	808	20	11	6	3
3340/3344	12	348	12	7	3	2
3350	30	555	27	15	8	4
3375	12	959	40	25	14	8
3380[3]	15	885	46	31	18	10
3380[4]	15	1770	46	31	18	10
3380[5]	15	2655	46	31	18	10

Track Capacity Determination

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. Use the following calculations 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, use 15 bytes to determine track capacity.

Space Calculation Formulas for Models 2305 through 3375

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

[2] Assumes without keys (KL = 0).

[3] 3380 single capacity Models A04, AA4, B04, AD4, BD4, AJ4, BJ4, and CJ2.

[4] 3380 double capacity Models AE4 and BE4.

[5] 3380 triple capacity Models AK4 and BK4.

Device	Blocks With Keys	Blocks Without Keys
3350	267+KL+DL	185+DL
3375	224+((KL+191)/32)(32)+ ((DL+191)/32)(32)	224+((DL+191)/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. **Note:** Track overflow is not available on DASD models 3375 through 3380.

3380 (All Models)

Each 3380 model track is divided into 1499 user data cells (with IBM standard R0) or 1515 user data cells (without an R0 record). A record can occupy from 16 to 1515 of these cells. The number of cells (*Space*) occupied by a record is a function of the key length (*KL*) and data length (*DL*) as specified in the count area of the record. The following formula applies to both blocked and unblocked records.

Space Calculation Formula for 3380

The space, in cells, occupied by a record can be calculated from the following formula:

$Space = C + K + D$

where:

$C = 8$

If $KL=0$, $K=0$

If $KL \neq 0$,

$K = 7 + \frac{KL + 12}{32}$, rounded up to an integral value.

$D = 7 + \frac{DL + 12}{32}$, rounded up to an integral value.

Track Capacity

A track can hold a given set of records providing that the sum of the *Space* values for all records is less than or equal to the maximum value.

The maximum value for the sum is 1499 if an IBM standard R0 record is provided and the summation of *Space* values does not include R0.

The maximum value for the sum is 1515 if the summation of *Space* values includes R0.

A standard end-of-file record has a *Space* value of 16.

If all records on a track are of equal *KL* and *DL*, each of which occupies *Space* cells, the maximum number of records which can fit on a track is:

$\frac{1515}{Space}$, rounded down to an integral value.

If an IBM standard R0 is provided and all the other records on a track are of equal *KL* and *DL*, each of which occupies *Space* cells, the maximum number of records (other than R0) which can fit on a track is:

$\frac{1499}{Space}$, rounded down to an integral value.

See *3380 DAS Reference Summary* for more information.

Track Capacity Calculation Examples for 3380 Models

Equal Length Records

The examples of track capacity calculation with equal length records use a data set consisting of 45,000 records of 80 bytes, both unblocked and blocked as 900 blocks of 4000 bytes each. All tracks use an IBM standard R0.

Unblocked Records

Using the calculations shown in "Space Calculation Formula for 3380" on page 215 with *KL*=0 and *DL*=80:

$C = 8$

$K=0$

$D=7+\frac{DL+12}{32}$, rounded up to an integral value,

$=7+\frac{80+12}{32}$, rounded up to an integral value,

$=10$

$Space=C+K+D$

$=8+0+10$

$=18$

The maximum number of records (other than R0) which can fit on a track is:

$\frac{1499}{Space}$, rounded down to an integral value.

$=\frac{1499}{18}$, rounded down to an integral value.

$=83$

At 83 records per track, the 45,000 record data set would require 543 tracks, or 36 cylinders and 3 tracks.

Blocked Records

Using the calculations shown in "Space Calculation Formula for 3380" on page 215 with *KL*=0 and *DL*=4000:

$C = 8$

$K=0$

$D=7+\frac{DL+12}{32}$, rounded up to an integral value,

$=7+\frac{4000+12}{32}$, rounded up to an integral value,

$=133$

$$Space = C + K + D$$
$$= 8 + 0 + 133$$
$$= 141$$

The maximum number of records (other than R0) which can fit on a track is:

$$\frac{1499}{Space}, \text{ rounded down to an integral value.}$$
$$= \frac{1499}{141}, \text{ rounded down to an integral value.}$$
$$= 10$$

At 10 blocks per track, the 900 block data set would require 90 tracks, or 6 cylinders.

Unequal Length Records

As an example of calculating track capacity for unequal length records, consider a partitioned data set with the following records:

1. A directory, consisting of 25 records, each with *KL*=8 and *DL*=256.
2. A number of members, blocked into 4096-byte blocks.
3. End-of-file records between blocks which contain different members.

All tracks contain an IBM standard R0.

Space Calculations

The space occupied by the different types of records is first calculated.

Directory Records: Using the calculations shown in "Space Calculation Formula for 3380" on page 215 with *KL*=8 and *DL*=256:

$$C = 8$$

$$K = 7 + \frac{KL + 12}{32}, \text{ rounded up to an integral value,}$$
$$= 7 + \frac{8 + 12}{32}, \text{ rounded up to an integral value,}$$
$$= 8$$

$$D = 7 + \frac{DL + 12}{32}, \text{ rounded up to an integral value,}$$
$$= 7 + \frac{256 + 12}{32}, \text{ rounded up to an integral value,}$$
$$= 16$$

$$Space = C + K + D$$
$$= 8 + 8 + 16$$
$$= 32$$

Member Blocks: Using the calculations shown in "Space Calculation Formula for 3380" on page 215 with *KL*=0 and *DL*=4096:

$C = 8$

$K=0$

$D=7+\frac{DL+12}{32}$, rounded up to an integral value,

$=7+\frac{4096+12}{32}$, rounded up to an integral value,

$=136$

$Space=C+K+D$

$=8+0+136$

$=144$

End-of-File Records: End-of-file records have a *Space* value of 16.

Track Capacity

First Track: In 3380 models, all tracks have an available *Space* value of 1499 cells. The directory consists of 25 records, each occupying 32 cells. Hence, $1499-(25 \times 32)=699$ cells are available for member blocks and end-of-file records. Each member block occupies 144 cells. Hence, in addition to the directory, the first track can contain 4 member blocks with 123 cells left over for end-of-file records. Since an end-of-File record occupies 16 cells, there is enough space for up to 4 end-of-file records which is the maximum required for 4 member blocks.

In this example, the first track of the data set would contain the directory and 4 or fewer member blocks, with end-of-file records as required.

Subsequent Tracks: The available *Space* of 1499 cells on subsequent tracks could hold 10 member blocks of 144 cells each. The remaining space of $1499-(10 \times 144)=59$ cells would hold up to 3 end-of-file records occupying 16 cells each. If more end-of-file records were necessary, the number of member blocks would have to be decreased to 9 to accommodate them.

In this example, tracks other than the first track of the data set would contain either:

10 member blocks and 3 or fewer end-of-file records OR

9 or fewer member blocks and 9 or fewer end-of-file records.

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.
OPEN/EOV nonspecific tape volume mount	17	Option to specify a tape volume serial number.

Entry Type	Hex Code	3-Byte Address—Purpose
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.

To dynamically shorten the list during execution, set the high-order bit of the word to a value of 1. To dynamically inactivate an entry in the list, set 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'—X'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 *DFP: Customization*.

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. You can select a control character option by coding the appropriate character in the **RECFM** operand of the DCB macro instruction. If either option is specified in the data control block, you must include a control character in each record. 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'5A'[1]	Change from line mode to page mode	
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'

[1] With the IBM 3800 Model 3 all-points-addressable mode, this code means that the record is in page mode instead of line compatibility mode.

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'81'	Select stacker 3

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, you can specify control characters defined by the International Organization for Standardization (ISO), American National Standards Institute (ANSI), or the Federal Information Processing Standards (FIPS). 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
X'5A'[1]	Change from line mode to page mode

[1] With the IBM 3800 Model 3 all-points-addressable mode, this code means that the record is in page mode instead of line compatibility mode.

Code	Action after Punching a Card
V	Select punch pocket 1
W	Select punch pocket 2

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.

Tables for Translating Codes

Translating from EBCDIC to ASCII

The next line shows that the first four EBCDIC values (00, 01, 02, 03) are not changed during translation to ASCII.

	0 1 2 3 4 5 6 7	8 9 A B C D E F
00-0F	000102031A091A7F	1A1A1A0B0C0D0E0F
10-1F	101112131A1A081A	18191A1A1C1D1E1F
20-2F	1A1A1A1A1A0A171B	1A1A1A1A1A050607
30-3F	1A1A161A1A1A1A04	1A1A1A1A14151A1A
40-4F	201A1A1A1A1A1A1A	1A1A5B2E3C282B21
50-5F	261A1A1A1A1A1A1A	1A1A5D242A293B5E
60-6F	2D2F1A1A1A1A1A1A	1A1A7C2C255F3E3F
70-7F	1A1A1A1A1A1A1A1A	1A603A2340273D22
80-8F	1A61626364656667	68691A1A1A1A1A1A
90-9F	1A6A6B6C6D6E6F70	71721A1A1A1A1A1A
A0-AF	1A7E737475767778	797A1A1A1A1A1A1A
B0-BF	1A1A1A1A1A1A1A1A	1A1A1A1A1A1A1A1A
C0-CF	7B41424344454647	48491A1A1A1A1A1A
D0-DF	7D4A4B4C4D4E4F50	51521A1A1A1A1A1A
E0-EF	5C1A535455565758	595A1A1A1A1A1A1A
F0-FF	3031323334353637	38391A1A1A1A1A1A

Translating from ASCII to EBCDIC

	0 1 2 3 4 5 6 7	8 9 A B C D E F
	00010203372D2E2F	1605250B0C0D0E0F
10-1F	101112133C3D3226	18193F271C1D1E1F
20-2F	404F7F7B5B6C507D	4D5D5C4E6B604B61
30-3F	F0F1F2F3F4F5F6F7	F8F97A5E4C7E6E6F
40-4F	7CC1C2C3C4C5C6C7	C8C9D1D2D3D4D5D6
50-5F	D7D8D9E2E3E4E5E6	E7E8E94AE05A5F6D
60-6F	7981828384858687	8889919293949596
70-7F	979899A2A3A4A5A6	A7A8A9C06AD0A107
80-8F	3F3F3F3F3F3F3F3F	3F3F3F3F3F3F3F3F
90-9F	3F3F3F3F3F3F3F3F	3F3F3F3F3F3F3F3F
A0-AF	3F3F3F3F3F3F3F3F	3F3F3F3F3F3F3F3F
B0-BF	3F3F3F3F3F3F3F3F	3F3F3F3F3F3F3F3F
C0-CF	3F3F3F3F3F3F3F3F	3F3F3F3F3F3F3F3F
D0-DF	3F3F3F3F3F3F3F3F	3F3F3F3F3F3F3F3F
E0-EF	3F3F3F3F3F3F3F3F	3F3F3F3F3F3F3F3F
F0-FF	3F3F3F3F3F3F3F3F	3F3F3F3F3F3F3F3F

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 I/O support function that took the exit.

Offset	Bytes and Alignment	Field Name	Description
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.

Data Control Block—BPAM, BSAM, QSAM

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	DCBBUFCA		Address of buffer pool control block.
24(18)	2	DCBBUFL		Length of buffer. May range from 0 to a maximum of 32760 bytes.
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.
		.100	S	Simple buffering.
		.110	A	QSAM locate mode processing of spanned records: OPEN is to construct a record area if it automatically constructs buffers.

Offset	Bytes and Alignment	Field Name	Description
		.010	R BSAM create BDAM processing of unblocked spanned records: Software track overflow. OPEN forms a segment work area pool. 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.
		.... 1...	XLRI being used to process a **RECFM=DS** or **RECFM=DBS** format tape data set (QSAM).
33(21)	.3	DCBEODA	End-of-data address. Address of a user-provided routine to handle end-of-data conditions.
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 routine to prevent overriding a specification of **KEYLEN=0** by a nonzero specification in the JFCB or data set label.

Offset	Bytes and Alignment	Field Name	Description
37(25)	.3	DCBEXLSA	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.
			Code
		Byte 1	<u>BSAM—Input</u>
		00..	Always zero for BSAM.
		..1.	R READ
		...x	Reserved bit.
		.... .1..	P POINT (which implies NOTE).
		.... ..1.	C CNTRL
51(33)		Byte 2	<u>BSAM—Output</u>
		00..	Always zero for BSAM
		..1.	W WRITE
		.... 1...	L Load mode BSAM (create BDAM data
		.... .1..	set).
		.... ..1.	P POINT
		1	C CNTRL BSAM create BDAM processing of unblocked spanned records, with **BFTEK=R** specified: The user's program has provided a segment work area pool.
50(32)		Byte 1	<u>QSAM—Input</u>
		.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.
51(33)		Byte 2	<u>QSAM—Output</u>
		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.

Offset	Bytes and Alignment	Field Name	Description
50(32)		Byte 1	
		00..	BPAM—Input Always zero for BPAM.
		..1.	R READ
		.... .1..	P POINT (which implies NOTE).
		...x x.xx	Reserved bits.
51(33)		Byte 2	
		00..	BPAM—Output 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.
17(11)	.1	DCBDEVT	Device type.
		1000 0011	3400 series magnetic tape unit.
		1000 1000	3480 Magnetic Tape Subsystem.
18(12)	..1	DCBDEN	Tape density—3400 series magnetic tape units.

Offset	Bytes and Alignment	Field Name	Description			
			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.
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 1101	Look at UCB DEVTYPE field or issue the DEVTYPE macro for the actual type of printer.

Offset	Bytes and Alignment	Field Name	Description
		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**.

Note: UCB DEVTYPE fields are presented in *Data Areas Volume 5*.

Access Method Interface

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.) Write-tape-immediate mode (3480). Window processing requested. (MSS)
		..1.	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	DCBSYNA	Address of user's synchronous error routine to be entered when a permanent error occurs.
62(3E)	..2	DCBBLKSI	Block size.
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.
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.

Offset	Bytes and Alignment	Field Name	Description
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.

QSAM 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.) Write-tape-immediate mode (3480). 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.
		.... .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	DCBSYNA	Address of the user's synchronous error routine to be entered when a permanent error occurs.
62(3E)	..2	DCBBLKSI	Block size.
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).

Offset	Bytes and Alignment	Field Name	Description
81(51)	.1	DCBBUFOF	Block prefix length.
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.
90(5A)	2	DCBPRECL	Block length, maximum block length, or data length.

Offset	Bytes and Alignment	Field Name	Description
16(10)	1	DCBKEYLE	Key length.
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 Disk Storage.
		0010 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	DCBBUFCA	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	DCBEODA	Address of a user-provided routine to handle end-of-data conditions.
36(24)	1	DCBRECFM	Record format.

Offset	Bytes and Alignment	Field Name	Description
			Code
		10..	F Fixed length records.
		01..	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	DCBEXLSA	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.
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
50(32)		Byte 1	BISAM
		00.0 0...	Always zero for BISAM.
		..1.	R READ
		.... .1..	S Dynamic buffering.
		.... ..1.	C CHECK
		x	Reserved bit.
51(33)		Byte 2	BISAM
		00.0 0000	Always zero for BISAM.
		..1.	W WRITE
50(32)		Byte 1	QISAM
		0.0. .0..	Always zero for BISAM.
		.1..	G GET
		...1	M Move mode of GET..
		.... 1...	L Locate mode for GET.
		xx	Reserved bit.

Offset	Bytes and Alignment	Field Name	Description
51(33)		Byte 2	QISAM
		1...	S SETL
		.1..	P PUT or PUTX
		..0.	Always zero for QISAM.
		...1	M Move mode of PUT.
		.... 1...	L Locate mode for 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 *Data Administration Guide*.
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.

Offset	Bytes and Alignment	Field Name	Description
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	DCBMSHIA	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).
		.... ..1.	Block could not be reached (BISAM only).
		1	Block could not be reached (input) (QISAM only).
			Overflow record (BISAM only).
			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.
150(96)	2	DCBNCRHI	Number of storage locations needed to hold the highest level index.
197(C5)	.1	DCBOVDEV	Device type for independent overflow.

Offset	Bytes and Alignment	Field Name	Description
		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.

Data Control Block—BDAM

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	DCBBUFCA	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:
		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.
		..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. 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.

Offset	Bytes and Alignment	Field Name	Description
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	DCBEXLSA	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 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.
50(32)		Byte 1	**Code**
		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.
51(33)		Byte 2	**Code**
		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.

Offset	Bytes and Alignment	Field Name	Description
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	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. A USING instruction specifying IHAPDAB and a dummy section base register containing the address of the actual parallel data access block should precede the use of any of the symbolic names provided by the dummy section.

```
                PDABD
IHAPDAB         DSECT
PDANODCB        DS      H   Number of DCB addresses in list
PDAMAXCB        DS      H   Maximum number of addresses allowed
                DS      A   Reserved for IBM use.
                DS      F   Reserved for IBM use.
PDADCBLA        DS      A   Address of last DCB entry
PDADCBEP        DS      A   Address of DCB entry last processed
                DS      F   Reserved for IBM use.
PDADCBAL        EQU     *   Start of DCB list
```

Abbreviations

The following abbreviations are defined as they are used in the MVS/DFP library. If you do not find the abbreviation you are looking for, see *Dictionary of Computing*, SC20-1699.

This list includes acronyms and abbreviations developed by the American National Standards Institute (ANSI) and the International Organization for Standardization (ISO). This material is reproduced from the *American National Dictionary for Information Processing*, copyright 1977 by the Computer and Business Equipment Manufacturers American National Standards Institute, 1430 Broadway, New York, New York 10018.

ABE. Abnormal end (value of EROPT).

ABEND. Abnormal end.

ABSTR. Absolute track (value of SPACE).

ACC. Accept erroneous block (value of EROPT).

ACS. Automatic class selection.

AFF. Affinity.

AL. American National Standard Labels.

ANSI. American National Standards Institute.

ASCII. American National Standard Code for Information Interchange.

AUL. American National Standard user labels (value of LABEL).

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).

BPAM. Basic partitioned access method.

BPI. Bits per inch.

BSAM. Basic sequential access method.

BSM. Backspace past tape mark and forward space over tape mark (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).

BUFL. Buffer length (operand of DCB).

BUFNO. Buffer number (operand of DCB).

BUFOFF. Buffer offset.

CCHHR. Cylinder/head record address.

CCW. Channel command word.

CNTRL. Control (macro instruction).

CONTIG. Contiguous space allocation (value of SPACE).

CSECT. Control section.

CSW. Channel status word.

CVOL. Control volume.

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. Direct access storage device.

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. Data control block.

DCBD. Data control block dummy section.

DD. Data definition.

DDNAME. Data definition name.

DEB. Data extent block.

DECB. Data event control block.

DEN. Magnetic tape density (operand of DCB).

DEVD. Device-dependent (operand of DCB).

DFDSS. Data Facility Data Set Services.

DISP. Data set disposition (parameter of DD statement).

DPI. Data protection image.

DS. ISCII/ASCII variable-length, spanned records (value of RECFM).

DSCB. Data set control block.

DSECT. Dummy control section.

DSNAME. Data set name.

DSORG. Data set organization (operand of DCB).

EBCDIC. Extended binary-coded decimal interchange code.

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).

ESTAE. Extended specify task abnormal exit.

EXCP. Execute channel program.

EXLST. 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.

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 tape mark and backspace over tape mark (operand of CNTRL).

FSR. Forward space over a specified number of blocks (records) (operand of CNTRL).

GCR. Group coded recording.

GDG. Generation data group.

GDS. Generation data set.

GL. GET macro, locate mode (value of MACRF).

GM. GET macro, move mode (value of MACRF).

HA. Home address.

IBG. Inter-block gap.

ID. Identifier.

INOUT. Input then output (operand of OPEN).

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. Kilobyte.

KEYLEN. Key length (operand of DCB).

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).

| **M**. Megabyte

MACRF. Macro instruction form (operand of DCB).

MBBCCHHR. Module#, bin#, cylinder#, head#, record#.

MOD. Modify data set (value of DISP).

MSHI. Main storage for highest-level index (operand of DCB).

MSS. 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).

NOPWREAD. No password required to read a data set (value of LABEL).

NRZI. Nonreturn-to-zero-inverted.

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).

OPTCD. Optional services code (operand of DCB).

OS CVOL. Operating system control volume.

OS/VS. Operating system/virtual storage.

OUTIN. Output then input (operand of OPEN).

PCI. Program-controlled interruption.

PDAB. Parallel data access block.

PDS. Partitioned data set.

PE. Phase encoding (tape recording mode).

PL. PUT macro, locate mode (value of MACRF).

PM. PUT macro, move mode (value of MACRF).

PO. Partitioned organization (value of DSORG).

POU. Partitioned organization unmovable (value of DSORG).

PRTSP. Printer line spacing (operand of DCB).

PS. Physical sequential (value of DSORG).

PSF. Print Services Facility.

PSU. Physical sequential unmovable (value of DSORG).

QISAM. Queued indexed sequential access method.

QSAM. Queued sequential access method.

R0. Record zero.

RACF. Resource Access Control Facility.

RDBACK. Read backward (operand of OPEN).

RDW. Record descriptor word.

RECFM. Record format (operand of DCB).

RKP. Relative key position (operand of DCB).

RLSE. Release unused space (DD statement).

RPS. Rotational position sensing.

SAM. Sequential access method.

SDW. Segment descriptor word.

SER. Volume serial number (value of VOLUME).

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).

| **SLI**. Suppress length indication bit.

SMS. Storage Management Subsystem.

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).

SS. Select stacker on card reader (operand of CNTRL).

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.

T. Track overflow option (value of RECFM); user-totaling (value of OPTCD).

TIOT. Task I/O table.

TRC. Table reference character.

TRTCH. Track recording technique (operand of DCB).

TSO. Time sharing option.

TTR. Track record address.

U. Undefined length records (value of RECFM).

UCS. Universal character set.

UHL. User header label.

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).

VS. Variable-length, spanned records.

VSAM. Virtual storage access method.

VTOC. Volume table of contents.

XLRI. Extended logical record interface.

Glossary

The following terms and abbreviations are defined as they are used in the MVS/DFP library. If you do not find the term or abbreviation you are looking for, see *Dictionary of Computing,* SC20-1699.

This glossary includes acronyms and abbreviations developed by the American National Standards Institute (ANSI) and the International Organization for Standardization (ISO). This material is reproduced from the *American National Dictionary for Information Processing*, copyright 1977 by the Computer and Business Equipment Manufacturers American National Standards Institute, 1430 Broadway, New York, New York 10018.

A

abnormal end (ABEND). Termination of a task prior to its completion as a result of an error condition that could not be resolved by error recovery facilities during task execution.

access method. A technique for organizing and moving data between main storage and input/output devices.

access method services. A multifunction service program that is used to manage both VSAM and non-VSAM data sets and integrated catalog facility or VSAM catalogs. Access method services is used to define data sets and allocate space for them, convert indexed-sequential (ISAM) data sets to key-sequenced data sets, modify data set attributes in the catalog, reorganize data sets, facilitate data portability between operating systems, create backup copies of data sets and indexes, help make inaccessible data sets accessible, list the records of data sets and catalogs, define and build alternate indexes, and convert OS CVOLs and VSAM catalogs to integrated catalog facility catalogs.

ACS routine. A procedural set of ACS language statements. Based on a set of input variables, the ACS language statements generate the name of a predefined SMS class, or a list of names of predefined storage groups, for a data set.

address marker. A byte of data on a disk or diskette, used to identify the data field and ID field in the record.

alias. An alternative name for an entry or for a member of a partitioned data set (PDS).

alias entry. An entry that relates an alias to the real entry name of a user catalog or non-VSAM data set.

allocation. Generically, the entire process of obtaining a volume and unit of external storage, and setting aside space on that storage for a data set.

application. The use to which an access method is put or the end result that it serves; contrasted to the internal operation of the access method.

automatic class selection (ACS). A mechanism for assigning SMS classes and storage groups to data sets.

auxiliary storage. All addressable storage, other than the memory of a processing unit, that can be accessed using an input/output channel; for example, storage on DASD, tape, or mass storage system volumes.

B

basic direct access method (BDAM). An access method used to directly retrieve or update particular blocks of a data set on a DASD.

basic partitioned access method (BPAM). An access method used to create program libraries on DASD for convenient storage and retrieval of programs.

basic sequential access method (BSAM). An access method for storing or retrieving data blocks in a continuous sequence, using either a sequential access or direct access device.

block prefix. An optional variable length field that may precede unblocked records or blocks of records in ASCII on magnetic tapes.

block size. The number of records, words, or characters in a block; usually specified in bytes.

blocking. The process of combining two or more records into one block.

buffer. A routine or storage used to compensate for a difference in the rate of flow of data, or time of occurrence of events, when transferring data from one device to another.

buffer pool. A continuous area of storage divided into buffers.

C

cache. In a storage controller, a high-speed storage buffer that is continually updated to contain recently-accessed contents of DASD storage. Its purpose is to reduce access time.

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 (SIO) instruction.

class. See *SMS class*.

configuration. (1) The arrangement of a computer system as defined by the characteristics of its functional units. (2) See *SMS configuration*.

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 later 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.

control section (CSECT). The part of a program specified by the programmer to be a relocatable unit, all elements of which are to be loaded into adjoining storage locations for execution.

control unit. A hardware device that controls the reading, writing, or displaying of data at one or more input/output devices. See also *storage control*.

control volume (CVOL). A volume that contains one or more indexes of the catalog.

cylinder. The tracks of a disk storage device that can be accessed without repositioning the access mechanism.

D

data class. A list of the data set allocation parameters and their values, used when allocating a new SMS-managed data set.

data control block (DCB). 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 (DEB). A control block that describes the physical attributes of the data set.

Data Facility Data Set Services (DFDSS). An IBM licensed program used to copy, move, dump, and restore data sets and volumes.

Data Facility Product (DFP). An IBM licensed program used to manage programs, devices, and data in an MVS operating environment.

data management. The task of systematically identifying, organizing, storing, and cataloging data in an operating system.

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-, variable-, or undefined-length records in auxiliary storage.

data set control block (DSCB). A control block in the VTOC that describes data set characteristics.

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.

dequeue. To remove items from a queue. Contrast with *enqueue*.

device address. Three or four hexadecimal digits that uniquely define a physical I/O device on a channel path in System/370 mode. The one or two leftmost digits are the address of the channel to which the device is attached. The two rightmost digits represent the unit address.

direct access. The retrieval or storage of data by a reference to its location in a data set rather than relative to the previously retrieved or stored data. See also *addressed-direct access* and *keyed-direct access*.

direct access device space management (DADSM). A DFP component used to control space allocation and deallocation on DASD.

direct access storage device (DASD). 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

MVS/XA. An MVS operating system environment that supports 31-bit real and virtual storage addressing, increasing the size of addressable real and virtual storage from 16 megabytes to 2 gigabytes.

MVS/370. An MVS operating system environment that supports 24-bit real and virtual storage addressing.

N

non-VSAM data set. A data set created and accessed using one of the following methods: BDAM, BPAM, BSAM, QSAM, QISAM.

O

online. Pertaining to equipment, devices, or data under the direct control of the processor.

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.

optimum block size. For non-VSAM data sets, optimum block size represents the block size that would result in the greatest space utilization on a device, taking into consideration record length and device characteristics.

OS control volume (OS CVOL). A volume that contains one or more indexes of the catalog.

P

page. (1) A fixed-length block of instructions, data, or both, that can be transferred between real storage and external page storage. (2) To transfer instructions, data, or both between real storage and external page storage.

page space. A system data set that contains pages of virtual storage. The pages are stored into and retrieved from the page space by the auxiliary storage manager.

paging. A technique in which blocks of data, or pages, are moved back and forth between main storage and auxiliary storage. Paging is the implementation of the virtual storage concept.

partitioned data set (PDS). A data set in DASD storage that is divided into partitions, called members, each of which can contain a program, part of a program, or data. See also *library*.

password. A unique string of characters stored in a catalog that a program, a computer operator, or a terminal user must supply to meet security requirements before a program gains access to a data set.

PDS directory. A set of records in a partitioned data set (PDS) used to relate member names to their locations on a DASD volume.

physical record. A record whose characteristics depend on the manner or form in which it is stored, retrieved, or moved. A physical record may contain all or part of one or more logical records.

pointer. An address or other indication of location.

primary space allocation. Initially allocated space on a direct access storage device, occupied by or reserved for a particular data set. See also *secondary space allocation*.

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, and programs written by a user.

Q

queued sequential access method (QSAM). An extended version of the basic sequential access method (BSAM). Input data blocks awaiting processing or output data blocks awaiting transfer to auxiliary storage are queued on the system to minimize delays in I/O operations.

R

record. A set of data treated as a unit.

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 (RACF). An IBM licensed program that provides access control by identifying and verifying users to the system. RACF authorizes access to DASD data sets, logs unauthorized access attempts, and logs accesses to protected data sets.

rotational position sensing (RPS). A function that permits a DASD to reconnect to a block multiplexer channel when a specified sector has been reached. This allows the channel to service other devices on the channel during positional delay.

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. The home address is written after the index point on each track.

I

indexed sequential access method (ISAM). An access method that retrieves or updates blocks of data using an index to locate the data set.

initial program load (IPL). (1) The initialization procedure that causes an operating system to commence operation. (2) The process by which a configuration image is loaded into storage at the beginning of a work day or after a system malfunction.

integrated catalog facility. The name of the catalog in MVS/DFP that replaces OS CVOLs and VSAM catalogs.

internal storage. Storage that is accessible by a computer without the use of input/output channels.

I/O device. An addressable input/output unit, such as a direct access storage device, magnetic tape device, or printer.

J

job control language (JCL). A problem-oriented language used to identify the job or describe its requirements to an operating system.

job entry subsystem (JES). A system facility for spooling, job queueing, and managing input and output. The two types of job entry subsystems in MVS are JES2 and JES3.

K

key. One or more characters within an item of data that are used to identify it or control its use. As used in this publication, one or more consecutive characters taken from a data record, used to identify the record and establish its order with respect to other records.

kilobyte. 1024 bytes.

L

library. A partitioned data set containing a related collection of named members. See *partitioned data set*.

load module. The output of the linkage editor; a program in a format ready to load into virtual storage for execution.

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, 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.

M

management class. A list of the migration, backup, and retention parameters and their values, for an SMS-managed data set.

Mass Storage System. The name for the entire storage system, consisting of the Mass Storage Facility and all devices that are defined to the Mass Storage Control.

mass storage volume. Two data cartridges in the IBM 3850 Mass Storage System that contain information equivalent to what would be stored on a direct-access storage volume.

master catalog. A catalog that contains extensive data set and volume information that VSAM requires 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 accumulate usage statistics for data sets.

megabyte (Mb). 1,048,576 bytes

member. A partition of a partitioned data set.

move mode. A transmittal mode in which the record to be processed is moved into a user work area.

MVS/DFP. An IBM licensed program which is the base for the Storage Management Subsystem.

MVS/ESA. An MVS operating system environment which supports ESA/370.

MVS/SP. An IBM licensed program used to control the MVS operating system and establish a base for an MVS/ESA, MVS/XA, or MVS/370 environment.

MVS/XA. An MVS operating system environment that supports 31-bit real and virtual storage addressing, increasing the size of addressable real and virtual storage from 16 megabytes to 2 gigabytes.

MVS/370. An MVS operating system environment that supports 24-bit real and virtual storage addressing.

N

non-VSAM data set. A data set created and accessed using one of the following methods: BDAM, BPAM, BSAM, QSAM, QISAM.

O

online. Pertaining to equipment, devices, or data under the direct control of the processor.

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.

optimum block size. For non-VSAM data sets, optimum block size represents the block size that would result in the greatest space utilization on a device, taking into consideration record length and device characteristics.

OS control volume (OS CVOL). A volume that contains one or more indexes of the catalog.

P

page. (1) A fixed-length block of instructions, data, or both, that can be transferred between real storage and external page storage. (2) To transfer instructions, data, or both between real storage and external page storage.

page space. A system data set that contains pages of virtual storage. The pages are stored into and retrieved from the page space by the auxiliary storage manager.

paging. A technique in which blocks of data, or pages, are moved back and forth between main storage and auxiliary storage. Paging is the implementation of the virtual storage concept.

partitioned data set (PDS). A data set in DASD storage that is divided into partitions, called members, each of which can contain a program, part of a program, or data. See also *library*.

password. A unique string of characters stored in a catalog that a program, a computer operator, or a terminal user must supply to meet security requirements before a program gains access to a data set.

PDS directory. A set of records in a partitioned data set (PDS) used to relate member names to their locations on a DASD volume.

physical record. A record whose characteristics depend on the manner or form in which it is stored, retrieved, or moved. A physical record may contain all or part of one or more logical records.

pointer. An address or other indication of location.

primary space allocation. Initially allocated space on a direct access storage device, occupied by or reserved for a particular data set. See also *secondary space allocation*.

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, and programs written by a user.

Q

queued sequential access method (QSAM). An extended version of the basic sequential access method (BSAM). Input data blocks awaiting processing or output data blocks awaiting transfer to auxiliary storage are queued on the system to minimize delays in I/O operations.

R

record. A set of data treated as a unit.

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 (RACF). An IBM licensed program that provides access control by identifying and verifying users to the system. RACF authorizes access to DASD data sets, logs unauthorized access attempts, and logs accesses to protected data sets.

rotational position sensing (RPS). A function that permits a DASD to reconnect to a block multiplexer channel when a specified sector has been reached. This allows the channel to service other devices on the channel during positional delay.

S

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 interval or on occurrence of a specified program interrupt.

secondary space allocation. A predefined contiguous space on a DASD volume reserved for additions to a particular data set, and allocated only after the primary allocation space is full. See also *primary space allocation.*

sequential access. The retrieval or storage of a data record in: its entry sequence, its key sequence, or its relative record number sequence, relative to the previously retrieved or stored record. See also *addressed-sequential access* and *keyed-sequential access.*

sequential access method (SAM). An access method for storing or retrieving data blocks in a continuous sequence, using either a sequential access or a direct access device.

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.*

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.

SMS class. A list of attributes that SMS applies to data sets having similar allocation (data class), performance (storage class), or backup and retention (management class) needs.

SMS-managed data set. A data set that has been assigned a storage class.

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.

spanned record. A logical record whose length exceeds control interval length, and as a result, crosses, or spans, one or more control interval boundaries within a single control area.

storage administrator. A person in the data processing installation who is responsible for defining, implementing, and maintaining storage management policies.

storage class. A list of DASD storage performance, security, and availability service level requirements for an SMS-managed data set.

storage group. A list of traits and characteristics that SMS applies to groups of storage volumes having similar migration, backup, and dump needs. Only the storage administrator can access storage group definitions.

Storage Management Subsystem (SMS). An operating environment that helps automate and centralize the management of storage. To manage storage, SMS provides the storage administrator with control over data class, storage class, management class, storage group, and ACS routine definitions.

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.

system-managed storage. An approach to storage management in which the system determines data placement and an automatic data manager handles data backup, movement, space, and security.

system residence volume (SYSRES). The volume on which the nucleus of the operating system and the master catalog are stored.

T

time sharing option (TSO). An optional configuration of the operating system that provides conversational time sharing from remote stations.

trailer label. A file or data set label that follows the data records on a unit of recording media.

U

unit address. The last two hexadecimal digits of a device address. This identifies the storage control and DAS string, controller, and device to the channel subsystem. Often used interchangeably with control unit address and device address in System/370 mode.

universal character set (UCS). A printer feature that permits the use of a variety of character arrays. Character sets used for these printers are called UCS images.

user catalog. An optional catalog used in the same way as the master catalog and pointed to by the master catalog. It also lessens the contention for the master catalog and facilitates volume portability.

V

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. A certain portion of data, together with its data carrier, that can be mounted on the system as a unit; for example, a tape reel or a disk pack. For DASD, a volume refers to the amount of space accessible by a single actuator.

volume table of contents (VTOC). A table on a direct access storage device (DASD) that describes each data set on the volume.

Index

A

A-type address constant
  defined 6
abbreviations
  list 247
ABEND macro
  BDAM 36
  BPAM 51
  BSAM 66
  list format 221—222
  QSAM 91
absexp defined 6
absolute expression defined 6
access methods
  general description
    BDAM 33
    BISAM 42
    BPAM 48
    BSAM 55
    QISAM 72
    QSAM 80
  macro instructions used with 194
  not recommended, list of xi
acronyms 247
ACSMETH operand
  SYNADAF macro 183
actual device addressing
  BDAM 39
  QISAM 77
adding to a data set
  BDAM 38, 203
  BISAM 44, 199
  BPAM 201
  BSAM 201, 203
  QISAM 107
  QSAM 108
address feedback
  current block position 142
  next block position 143
address of buffers
  obtained from a pool 111
  returned to pool 104, 105
addressing
  types of (BDAM) 39
  24- and 31-bit modes 1
aids, coding 4
alias names
  in directory 178—179
alignment of buffers
  BDAM 33
  BISAM 42
  BPAM 48
  BSAM 56
alignment of buffers *(continued)*
  QISAM 72
  QSAM 81
alignment of printer forms
  automatic 156
  manual 161
allocation retrieval list 221
ANSI control characters
  BPAM 53
  BSAM 70
  QSAM 95
argument, search
  BDAM 38
  QISAM 76
ASCII data sets
  block prefix
    BSAM 58
    QSAM 84
    restriction 58, 84
  block size
    BSAM 57
    QSAM 82
  buffer length
    BSAM 58
    QSAM 83
  record format restriction
    BSAM 70
    QSAM 96
ASCII translation routines
  check routine 20
  get routine 108
  put routine 138
  write routine 201
  XLATE macro 208
associated data sets
  closing 23
  opening 119
  specifying
    BSAM 63, 65
    QSAM 88, 89
ATTACH macro
  relationship with BLDL macro 9
automatic buffer pool construction
  BDAM 33
  BISAM 42
  BPAM 48
  BSAM 55
  QISAM 72—74
  QSAM 80
automatic checkpoint restart 22
automatic error options
  See EROPT
automatic volume switching
  FEOV macro 101

B

backspacing
  BSP macro 12
  CNTRL macro 30
backward read
  open option 120
  read operation 147
base registers
  dummy sections 98
  macro instructions 6
basic direct access method
  See BDAM
basic partitioned access method
  See BPAM
basic sequential access method
  See BSAM
BDAM (basic direct access method)
  general description 33
  macro instructions used with 211
  symbolic field names in DCB 242—244
BDW (block descriptor word)
  BLKSIZE operand 49, 57
BFALN operand
  DCB macro
    BDAM 33
    BISAM 42
    BPAM 48
    BSAM 56
    QISAM 72
    QSAM 81
BFTEK operand
  DCB macro
    BDAM 34
    BSAM 56
    QSAM 81—82
BISAM (basic indexed sequential access method)
  general description 42
  macro instructions used with 211
  symbolic field names in DCB 238—242
BLDL macro
  described 9—11
  reason codes 11
  return codes 11
  use by access method 211
  using FIND macro 102
BLKSIZE operand
  DCB macro
    BDAM 34
    BPAM 49
    BSAM 56
    QISAM 73
    QSAM 82—83
block
  backspacing 12
  BUFOFF operand
    effect on buffer length 58
  count exit
    BSAM 66
    list format 221
block *(continued)*
  count exit *(continued)*
    QSAM 91
  data control 33
  data event control 209
  descriptor word
    BLKSIZE operand 49, 57, 73, 82
    BUFOFF operand 59, 84
    LRECL operand 76
  event control 209
  position feedback 131, 197
  positioning with POINT 131—132
  prefix
    See also BUFOFF operand
    effect on block length 57
    effect on buffer length 83
    effect on data alignment 56, 81
  reading 142—148
  size
    See BLKSIZE operand
  writing 196—204
block size
  for SYSOUT data sets
    See also BLKSIZE operand
    QSAM 82
  SYSOUT data sets
    BSAM 57
  system-determined
    BPAM 49
    BSAM 57
    QSAM 83
blocking
  data checks (UCS printer) 164
  records
    BDAM 33, 40
    BPAM 48, 53
    BSAM 70
    QISAM 78
    QSAM 95—97
boundary alignment
  See BFALN operand
BPAM (basic partitioned access method)
  general description 48
  macro instructions used with 211
  symbolic field names for DCB 229—236
BSAM (basic sequential access method)
  general description 55
  macro instructions used with 211
  symbolic field names for DCB 227—233
BSP macro
  described 12
  reason codes 13
  return codes 13
  use by access method 211
BUFCB operand
  DCB macro
    BDAM 35
    BISAM 43
    BPAM 58

BUFCB operand *(continued)*
DCB macro *(continued)*
BSAM 58
GETBUF macro 111
GETPOOL macro 112
QISAM 73
QSAM 83
buffer
alignment
See BFALN operand
BUFCB operand
GETPOOL macro 112
control
FREEBUF macro 104
FREEDBUF macro 105
FREEPOOL macro 106
GETBUF macro 111
GETPOOL macro 112
RELSE macro 153
forms control
SETPRT macro 156
length
ASCII data sets 58, 83
BUILD macro 15
BUILDRCD macro 16
card image mode 58, 83
GETPOOL macro 112
message format (SYNADAF macro) 185—186
pool construction
See also BUFCB operand
BUILD macro 14—15
BUILDRCD macro 16—17
releasing
FREEBUF macro 104
FREEDBUF macro 105
FREEPOOL macro 106
RELSE macro 153
SYNADRLS macro 187
specifying number
See BUFNO operand
buffering
dynamic 105
problem program controlled
BISAM 42
BSAM 55
simple 81—82
specifying 35, 56—57, 81—82
variable-length spanned record
BDAM 34
BSAM 57
BUILDRCD macro 16—17
QSAM 81
BUFL operand
DCB macro
BDAM 35
BISAM 43
BPAM 50
BSAM 58
QISAM 73
BUFL operand *(continued)*
DCB macro *(continued)*
QSAM 83
BUFNO operand
DCB macro
BDAM 35
BISAM 43
BPAM 50
BSAM 58
CNTRL macro 30
NCP operand 43
QISAM 74
QSAM 84
BUFOFF operand
DCB macro
BSAM 58—59
QSAM 84
effect on buffer length 58
BUILD macro
described 14—15
relationship to
BFALN operand 33
BUFCB operand 35
BUFL operand 35
BUFNO operand 35
use by access method 211
BUILDRCD macro
execute form 19
GET macro 109
list form 18
relationship to
BUFL operand 84
BUFNO operand 80
PUT macro 139
TRUNC macro 193
standard form 14—15
use by access method 211
BURST operand (SETPRT macro) 158, 173, 175
bypassing automatic forms alignment 161

C

capacity record (R0)
relationship with
READ macro 143
WRITE macro 196, 203
card codes
BSAM 62
QSAM 86
card image
buffer length required 58, 83
defined 62, 86
eliminate mode, read column
BSAM 62, 64
QSAM 87
card punch
described 62, 86

card reader
described 63, 87
carriage control channel
CNTRL macro 30—32
PRTOV macro 134—135
carriage control characters
CNTRL macro 30—32
machine 223—224
PRTOV macro 134—135
chained scheduling
BPAM 53
BSAM 68
QSAM 93
changing
name
partitioned data set 178
channel
overflow 134—135
programs, number
BPAM 52
BSAM 67
programs, number of
BISAM 45
channel programs
using suppress length indication (SLI) 159
character arrangement table
specifying use of 158
character set code
1403 printer 164
3203 printer 164
3211 printer 164
3262 Model 5 printer 164
4245 printer 164
4248 printer 164
CHARS operand (SETPRT macro) 158
CHECK macro
described 20
end of data (EODAD) 51, 65
operations (NCP) 52, 67
relationship to
MACRF operand 38
operations (NCP) 45
POINT macro 132
READ macro 142, 145, 147, 149
SYNCDEV macro 188
WRITE macro 196, 199, 201, 203
use by access method 211
checking, write-validity
BDAM 39
BPAM 53
QISAM 78
QSAM 94
checkpoint records, embedded (DOS)
CNTRL macro 30
POINT macro 131
CHKPT macro
use by access method 211
CLOSE macro
execute form 28
FREEPOOL macro 106
list form 26
MODE 25, 27, 28
relationship to
BUILDRCD macro 16
POINT macro 131
PUT macro 139
SETL macro 154
return codes 29
standard form 23—25
TYPE=T 24
use by access method 211
CNTRL macro
described 30—32
MACRF operand (DCB macro)
BSAM 67
QSAM 92
restrictions 30
use by access method 211
codes
See also card codes
See also completion codes
See also control characters
See also conversion
See also exception code
See also return codes
card
BSAM 62
QSAM 86
completion
See codes, return
control character
See control characters
conversion
ASCII to EBCDIC 20, 108, 208
EBCDIC to ASCII 201, 208
XLATE macro 208
return
BLDL macro 11
BSP macro 13
FIND macro 102
MSGDISP macro 116
NOTE macro 118
POINT macro 133
RELEX macro 152
SETPRT macro 164—169
STOW macro 180—181
SYNADAF macro 184
SYNADRLS macro 187
WRITE macro 205
coding
aids 4
macro instructions 1—7
registers as operands 6
column, binary
See also card image

column, binary *(continued)*
eliminate mode, read column
QSAM 88
completion codes
BLDL macro 11
BSP macro 13
FIND macro 102
MSGDISP macro 116
NOTE macro 118
POINT macro 133
RELEX macro 152
STOW macro 180—181
SYNADAF macro 184
SYNADRLS macro 187
WRITE macro 205
completion testing of I/O operations 20, 194
condition, exception 209
construct
a DECB (data event control block) 209
control
I/O device 30—32, 134
page format 134—135
printer (3800) 156—164
releasing
buffer pool (FREEPOOL macro) 106
buffer (FREEBUF macro) 104
data block (RELEX macro) 152
dynamically acquired buffer 105, 197
QSAM buffer (RELSE macro) 153
requesting
buffer (GETBUF macro) 111
data block 112
control characters
CNTRL macro 30—32
described 223—224
ISO/ANSI/FIPS 225
machine 223
PRTOV macro 134—135
specifying
BPAM 53
BSAM 70
QSAM 95
control section
See DCB macro
COPIES operand
SETPRT macro
modifying 164
specifying 159
copy modification module
specifying 162
COPYNR operand in SETPRT macro
modifying 164
specifying 159
COPYP parameter
described 159
execute form 176
list form 174
restrictions 159
COPYP parameter *(continued)*
3262 Model 5 printer 159
4248 printer 159
count exit, block
BSAM 66
format list 221
QSAM 91
cylinder
DCB macro 74
index 77
overflow area 74
CYLOFL operand (DCB macro)
described 74

D

D-format records
BSAM 70
QSAM 95
DASD (direct access storage device)
capacity 213—215
considerations with
BSP macro 12
CLOSE macro 23, 24
POINT macro 131—132
interface in DCB 232
data block
exclusive control of 143
locating with POINT macro 131—132
release of exclusive control 152
retrieving 107—110, 142—149
writing 136—141, 196—204
data check
blocking and unblocking 94, 163
restriction with CNTRL macro 30
data control block
See DCB
data definition (DD) statement
See DD statement
data event control block
See DECB
data extent block
See DEB
data management
parameter list 26, 125
data mode processing
GET macro 93, 110
PUT macro 93, 139
data protection image
See DPI
data set
block size for SYSOUT 57, 82
closing 23—25
connecting 119—123
disconnecting from 23—25
disposition at close 24
opening 119—123
organization
See DSORG operand

data set *(continued)*
temporary closing 25
types
See access methods
data translation
See code conversion
data transmittal modes
data 93, 110, 139
locate 107, 109, 136, 139
move 107, 109, 137, 139
specified in DCB 93
data, end of
See EODAD operand
DCB macro
BDAM 33—41
BISAM 42—47
BPAM 48—54
BSAM 55—71
QISAM 79
QSAM 81—97
use by access method 211
DCB open exit routine
OPTCD operand 39, 69
restriction with BUILDRCD macro 16
DCB operands
described
See DCB macro
symbolic names for 227—244
DCB (data control block)
abend exit
BDAM 36
BPAM 51
BSAM 66
list format 221—222
QSAM 91
completing 119
data event 145—149, 209
DCBEXCD1 field 209
DCBEXCD2 field 209
DCBLRECL field 138
DCBNCRHI field 46
DCBOFLGS field 122
described
See DCB macro
dummy section for 98—99
exit list
See EXLST operand
special options with BLDL macro 9—11
specifying operands 26
symbolic references to 227—244
DCBD macro
described 98—99
use by access method 211
DD statement
See also DDNAME operand
DCB 117, 119, 121
NOTE macro 117
OPEN macro 119—121
DD statement *(continued)*
POINT macro 131
relationship to data control block 1, 131
DDNAME operand
DCB macro
BDAM 36
BISAM 43
BPAM 50
BSAM 59
QISAM 74
QSAM 84
deblocking records
BDAM 33, 40
BPAM 53
BSAM 70
QISAM 78
DECB (data event control block)
construction 150, 206
described 209
exception code 209
modifying with execute form 151, 207
requirement with CHECK macro 20
requirement with FREEDBUF macro 105
delete option
described 77
DEN operand
DCB macro
BSAM 60
QSAM 85
density, recording
See DEN operand
descriptor word
block
BPAM 49
BSAM 57, 59, 149
QISAM 73, 76
QSAM 82, 84
record
BSAM 59
QISAM 73, 76
QSAM 91
segment 57, 149
DEVD operand
DCB macro
BSAM 59—65
DCBD macro 99
QSAM 84—89
device addressing, types of (BDAM) 39
device capacities 213—215
device types in a dummy section 99
direct data set
See BDAM data set
direct search option
QSAM 94
directory
obtaining contents with BLDL
partitioned data set 9—11
partitioned data set
operations performed by STOW
macro 178—179

directory *(continued)*
partitioned data set *(continued)*
search by FIND macro 102
DISP option
See disposition option
disposition option
CLOSE macro 24
OPEN macro 121
requirement for extending an ISAM data set 117
DOS (disk operating system)
embedded checkpoint records
CNTRL macro 30
DOS/OS interchange feature, specifying 69, 95
POINT macro 131
doubleword alignment
See BFALN operand
DPI (data protection image)
BSAM 63, 64
QSAM 88, 89
DSECT statement
DCB symbolic names 227
DSORG operand
DCB macro
BDAM 36
BISAM 44
BPAM 50
BSAM 65
QISAM 75
QSAM 90
DSORG operand (CHECK macro) 21
dummy control section
DCBD macro 98—99
PDABD macro 130
used for DCB 227
dummy data block (BDAM) 203—204
dummy key 203
dynamic buffering
effect on buffer length 35, 42
effect on number of channel programs 45
requesting in READ macro 143, 146
requesting in WRITE macro 197, 199
returning buffer to pool 105, 197
specified in BDAM DCB 38
specified in BISAM DCB 45

E

EBCDIC (extended binary coded decimal interchange code)
ASCII translation
check routine 20
DCB option 68, 94
GET routine 108
put routine 138
write routine 201
XLATE macro 208
ECB operand
WAIT macro 194
ECB (event control block)
described 209
ECBLIST operand
WAIT macro 194
eliminate mode, read column
BSAM 64
QSAM 87, 88
embedded checkpoint records (DOS)
CNTRL macro 30
POINT macro 131
end-of-data
See EODAD operand
end-of-data routine
See EODAD routine
end-of-file on magnetic tape
ignored
BSAM 69
QSAM 95
end-of-sequential retrieval
See ESETL
entry
to exit routine 221
to SYNAD exit routine 209
EODAD operand
DCB macro
BPAM 51
BSAM 65
QISAM 75
QSAM 90
EODAD (end-of-data) routine
BSP macro 12
CHECK macro 20
CNTRL macro 30
FEOV macro 101
GET macro 107, 110
POINT macro 132
EOV (end-of-volume)
exit
BSAM 66
QSAM 91
forced (FEOV macro) 101
EROPT (automatic error options) operand
DCB macro 90—91
ERP (error recovery procedure)
QSAM 94
error analysis, I/O
DCB macro 69
BSAM 69
GET macro 107, 110
relationship with
CHECK macro 20
CNTRL macro 31—32
DCB macro 94
POINT macro 132
PUT macro 137, 140
PUTX macro 141
SETL macro 155
SYNADAF macro 182

error analysis, I/O *(continued)*
specifying in DCB macro
BDAM 40
BISAM 46
BPAM 54
QISAM 79
QSAM 96
status indicators
QISAM 209
error codes
See return codes
error conditions
opening a data set 123
error exits
CHECK macro 20
CNTRL macro 31—32
DCB macro 69, 94
GET macro 107, 110
POINT macro 132
PUT macro 137, 140
PUTX macro 141
SETL macro 155
SYNADAF macro 182—183
error option operand (QSAM) 90
error recovery
procedure
tape 69, 94
ESETL (end-of-sequential retrieval) macro
described 100
GET macro 107
relationship to
SETL macro 154
use by access method 211
event control block
See ECB
open
See DCB open exit routine
exception code 209
exclusive control of data block (BDAM)
releasing 197
requesting 143
specified in DCB 39
EXCP macro
relationship with SYNADAF macro 182
execute form
BUILDRCD macro 19
CLOSE macro 28
OPEN macro 127
READ macro 151
SETPRT macro 175—177
WRITE macro 207
executing macros
addressing mode, 24- and 31-bit 1
exit routine
See also EXLST operand
block count 91
data control block
See EXLST operand
exit routine *(continued)*
end-of-data
See EODAD operand
end-of-volume 91
error analysis
See error exits
FCB image 91
list format 221
user labeling 91
user totaling 91
EXLST operand
DCB macro
BDAM 36
BISAM 44
BPAM 51
BSAM 65
list format 221
QISAM 75
QSAM 91
exit routine
block count 66
end-of-volume 66
FCB image 66
user labeling 66
user, processing 66
expressions
absolute (absexp) 6
relocatable (relexp) 6
EXTEND operand
OPEN macro 120
extended binary coded decimal interchange code
See EBCDIC
extended logical record interface
See XLRI
extended search option
LIMCT operand 37—38
OPTCD operand 39

F

F-format records
See RECFM operand
FCB (forms control buffer)
image
defining 66, 91
operand (SETPRT macro) 160
3203 printer 160
3211 printer 160
3262 Model 5 printer 160
4245 printer 160
4248 printer 160
feedback
block position 143, 197
next address 144
FEOV macro
use by access method 211
using 101

file, end of
  See end-of-file
FIND macro
  described 102
  reason codes 103
  return codes 102
  use by access method 211
fixed-length records
  See BLKSIZE operand, RECFM operand
FLASH operand
  modifying in SETPRT macro 164
  specifying SETPRT macro 161
format
  exit list 221
  page 134
  record
    BDAM 40
    BPAM 53
    BSAM 70—71
    QISAM 78
    QSAM 95—97
forms alignment 156, 160
forms control buffer
forms overlay frame 161
forward space (CNTRL macro) 31
FREE option
  CLOSE macro 24
FREEBUF macro
  described 104
  GETBUF macro 111
  relationship to
    BUILD macro 14
  use by access method 211
FREEDBUF macro
  described 105
  use by access method 211
  used with BISAM 44, 200
FREEPOOL macro
  described 106
  GETPOOL macro 112
  relationship to
    CLOSE macro 23
  use by access method 211
full-track-index write option 78
fullword boundary alignment
  See BFALN operand
FUNC operand
  DCB macro
    BSAM 62—64
    QSAM 87—89

G

GET macro
  ASCII translation 108
  data mode
    QSAM 93, 110
  locate mode
    QISAM 76, 107
    QSAM 93, 109
GET macro *(continued)*
  move mode
    QISAM 76, 107
    QSAM 93, 109
    restriction when using CNTRL macro 30, 92
  PDAB macro 129
  QISAM 107
  QSAM 108—110
  relationship to
    CNTRL macro 30
    EODAD (see EODAD operand) 1
    RELSE macro 153
    SETL macro 154
  specified in DCB macro
    QISAM 76
    QSAM 92
  TYPE=P 110
  use by access method 211
GET routine exits 107, 110
GETBUF macro
  described 111
  FREEBUF macro 104
  relationship to
    BUILD macro 14
  use by access method 211
GETPOOL macro
  described 112
  FREEPOOL macro 106
  relationship to
    BFALN operand 33
    BUFCB operand 35
    BUFL operand 35
    BUFNO operand 36
  use by access method 211
glossary 251, 257

I

IBM 3800 Printing Subsystem 156
IHADCB dummy section 98
IHAPDAB dummy section 130
image
  data protection
    BSAM 63, 65
    QSAM 88, 89
  FCB (forms control buffer) 66, 91, 160
  UCS (universal character set) 164
image mode, card
  BSAM 62
  QSAM 86
independent overflow area
  described 77
index
  cylinder 77
  highest-level
    address of 45
    size of 46
  master
    number of tracks per level 77
    specified in OPTCD operand (DCB macro) 77

INIT operand (SETPRT macro) 161
INOUT operand (OPEN macro) 120
input data set
closing 24—25
opening 119—123
QSAM 110
READ or GET specified in DCB
BDAM 38
BISAM 44
BPAM 52
BSAM 67
QISAM 76
QSAM 92
reading
BDAM 142—144
BISAM 144—145
BPAM 147—148
BSAM (read a direct data set) 149
BSAM (read a sequential data set) 147—148
QISAM 107
QSAM 108
testing completion of I/O operations
CHECK 20
WAIT 194—195
used with GET macro 108
INPUT operand (OPEN macro) 120
INPUT option
OPEN macro 119
input/output devices
card reader and card punch 30
control
PRTOV macro 134
control of
CNTRL macro 30—32
magnetic tape 30
printer 30
3505 card reader
DCB macro 64, 87, 88
3525 card punch
CLOSE macro 23
CNTRL macro 30
DCB macro 64, 65, 87, 88
OPEN macro 119
input/output error analysis
See SYNAD exit routine
input/output operations
completion of 20, 194
interface
DCB
BPAM 235—236
BSAM 235—236
card reader, card punch 233
direct access devices 232
magnetic tape 232
printer 233
QSAM 236
ISAM (indexed sequential access method)
See also BISAM, QISAM
ISAM (indexed sequential access method) *(continued)*
general description 42, 72
macro instructions used with 211
symbolic field names in DCB 238—242
ISO/ANSI/FIPS control characters
defined 225

J

JCL (job control language)
DD statement
CLOSE macro 23
data control block (see DDNAME operand) 1
DCB macro 36, 50
GET macro 108
NOTE macro 117
OPEN macro 119—120
POINT macro 131
PUT macro 136
LABEL parameter to request ASCII translation 20, 108, 138
JFCBE (job file control block extension)
exit list format 221
EXLST operand 91
OPTCD parameter 69
job step
checkpoint restart 22

K

key position, relative (RKP) 79
key (BDAM)
address 143
reading 142
specifying as search argument 38
specifying length 37
writing 197
key (ISAM)
address 146, 200
reading 146
specifying length 75
specifying position 79
writing 199
KEYLEN operand
DCB macro
BDAM 37
BPAM 51
BSAM 66
QISAM 75
key, record
PUT macro 136
READ macro 145
RKP (relative key position) operand 79
SETL macro 154—155
WRITE macro 200

L

LABEL operand
  DD statement 20, 108, 138
labels
  See also EXLST operand
  exit list format 221
  input data set 95, 101, 119
  output data set
    CLOSE macro 23
    FEOV macro 101
    OPEN macro 119
  user, processing 91
LEAVE option
  CLOSE macro 24
  FEOV macro 101
  OPEN macro 121
levels of master index (ISAM) 77
LIMCT operand
  DCB macro 37—38
line spacing, printer
  CNTRL macro 30—32
  PRTSP operand (DCB macro)
    BSAM 61
    QSAM 86
LINK macro
  relationship with BLDL macro 9
list address
  data management 28, 127, 178
list form
  BUILDRCD macro 18
  CLOSE macro 26
  OPEN macro 125
  READ macro 150
  SETPRT macro 173—174
  WRITE macro 206
list format, exit 221
LOAD macro
  relationship with BLDL macro 9
loading
  FCB (forms control buffer) 160
  UCS (universal character set buffer) 164
locate mode
  BUILDRCD macro 16
  GET macro
    QISAM 107, 109
    QSAM 109
  PUT macro
    QISAM 137
    QSAM 139
  specified in DCB macro
    QISAM 76
    QSAM 93
logical record interface
  See LRI
logical record length
  See also LRECL operand
  GET macro 108
  PUT macro 136, 138
logical record length *(continued)*
  PUTX macro 141
LONG operand
  WAIT macro 195
lower limit of sequential retrieval
  SETL macro 154—155
LRECL operand
  DCB macro
    BPAM 52
    BSAM 66
    QISAM 75
    QSAM 91
LRI (logical record interface)
  invoked by BUILDRCD macro 16
  provided by QSAM 82
  specifying in DCB macro (BFTEK) 81—82
  used with PUT macro 138

M

machine control characters
  BPAM 53
  BSAM 70
  described 223—224
  QSAM 95
MACRF operand
  DCB macro
    BDAM 38—39
    BISAM 44
    BPAM 52
    BSAM 67
    QISAM 76
    QSAM 92
  locate
    QISAM 107
  optical mark read
    BSAM 64
  read column eliminate
    BSAM 64
    QSAM 87
macro
  coding 1—4
  DCB
    BDAM 33—41
    BISAM 42—47
    BPAM 48—54
    BSAM 55—71
  GET
    QISAM 107
    QSAM 108—110
  OPEN
    execute form 127
    list form 125
    standard form 119—123
  return codes 29, 123
macros
  not recommended, list of xi

macros that run in 31-bit mode
  CLOSE 25
  OPEN 122
macros, data management
  BISAM 144
  BLDL 9
  BSP 12
  BUILD 14—15
  BUILDRCD
    execute form 19
    list form 18
    standard form 16—17
  CHECK 20
  CHKPT 22
  CLOSE
    execute form 28
    list form 26
    standard form 23—25
  CNTRL 30—32
  DCB
    QISAM 72—79
    QSAM 81—97
  DCBD 98—99
  ESETL 100
  FEOV 101
  FIND 102
  FREEDBUF 105
  FREEPOOL 106
  GETBUF 111
  GETPOOL 112
  NOTE 117
  PDAB 129
  PDABD 130
  POINT 131—132
  PRTOV 134—135
  PUT
    QISAM 136
    QSAM 138—139
  PUTX 141
  READ
    BDAM 142—144, 149
    BISAM 145
    BPAM 147—148
    BSAM 147—149
    execute form 151
    list form 150
  RELEX 152
  RELSE 153
  SETL 154—155
  SETPRT
    execute form 177
    list form 173—174
    standard form 156—172
  STOW 178—181
  SYNADAF 182—186
  SYNADRLS 187
  TRUNC 193
  using by access method 211
macros, data management *(continued)*
  WAIT 194—195
  WRITE
    BDAM 196—198, 203—205
    BISAM 199—200
    BPAM 201
    BSAM 201
    execute form 207
    list form 206
  XLATE 208
magnetic tape
  backspace
    BSP macro 12
    CNTRL macro 30
  considerations with
    BSP macro 12
    CLOSE macro 23—25
    POINT macro 131—132
  density 60, 85
  end-of-file, ignored 69, 95
  FEOV macro
    final volume positioning 101
  forward space 30
  interface in DCB 232
  read backward 147
  recording technique 60, 85
  restriction
    NOTE macro 117
    POINT macro 131
  short error recovery procedure 69, 94
Mass Storage System
  See MSS
master index
  highest level in storage
    address of storage area 45
    size of storage area 46
  number of tracks per level 77
  option specified in DCB 77
MAXDCB operand
  PDAB macro 129
member
  complete list with BLDL macro
    partitioned data set 9—11
  partitioned data set
    locate beginning with FIND macro 102
    update directory with STOW macro 178—179
MF operand
  BUILDRCD macro 18, 19
  CLOSE macro 28
  OPEN macro 125, 127
  READ macro 150, 151
  SETPRT macro 174, 177
  WRITE macro 206, 207
mode
  See also MACRF operand
  card image
    BSAM 62
    QSAM 86

mode *(continued)*
data
QSAM 93, 110, 139
locate
QISAM 76, 137
QSAM 93, 109, 139
move
QISAM 76, 107, 136
QSAM 93, 109, 139
optical mark read
QSAM 88
read column eliminate
QSAM 88
scan (QISAM) 77
MODE operand
DCB macro
BSAM 62, 63
QSAM 86, 88
MODIFY operand (SETPRT macro) 162
modifying
parameter list
BUILDRCD macro 19
CLOSE macro 28
OPEN macro 125, 127
READ macro 151
WRITE macro 207
modifying a parameter list
SETPRT macro 175
move mode
QISAM
GET macro 107
PUT macro 136
specified in DCB 76
QSAM
GET macro 109
PUT macro 139
specified in DCB 93
restriction 30, 93
MSGDISP macro
return codes 116
MSHI operand (DCB macro) 45
MSS (Mass Storage System)
DCB 232, 235, 236
MSWA operand
DCB macro 45
multiline print option
BSAM 63, 64
QSAM 87, 89

N

NCP operand
DCB macro
BISAM 45
BPAM 52
BSAM 67
next address feedback
BDAM (creating) 204
BDAM (existing) 143
not recommended
access methods
BDAM 33
BISAM 42
QISAM 72
ESETL macro 100
FREEDBUF macro 105
GET (QISAM) macro 107
PUT (QISAM) macro 136
READ (BDAM) macro 142, 149
READ (BISAM) macro 145
RELEX macro 152
SETL macro 154
WRITE (BDAM) macro 196, 203
WRITE (BISAM) macro 199
notational conventions
described 2
NOTE macro
described 117
POINT macro 117
restriction
BSP macro 12
return codes 118
specified in DCB
BPAM 52
BSAM 67
use by access method 211
NTM operand
DCB macro 78
number of channel programs
See NCP operand
number of tracks per index level
See NTM operand

O

OMR (optical mark read) mode
BSAM 64
QSAM 88
online printer
control 30—32
skipping 134, 223—224
spacing 134, 223—224
open exit
See DCB open exit routine
OPEN macro
execute form 127
FEOV macro 101
GETPOOL macro 112
list form 125
MODE 122, 126, 127
NOTE macro 117
relationship to
CLOSE macro 23
DDNAME operand 1
READ macro 147
WRITE macro 201
return codes 123
standard form 119—123

OPEN macro *(continued)*
TYPE 121, 126, 127
use by access method 211
open operation, testing 121—123
open options 119—121
operands
substitution for 4
OPTCD operand
DCB macro
BDAM 39
BISAM 45
BPAM 53
BSAM 69
QISAM 77
QSAM 93—95
SETPRT macro 163
optical mark read mode
See OMR
option codes
See OPTCD operand
organization, data set
See access methods
OUTIN operand (OPEN macro) 120
OUTINX operand (OPEN macro) 120
output data set
closing 23—25
opening 119—123
WRITE or PUT specified in DCB macro
BDAM 38
BISAM 44
BPAM 52
BSAM 67
QISAM 76
QSAM 93
writing
BDAM 196—198
BISAM 199—200
BPAM 201
BSAM (create a direct data set) 203—204
BSAM (sequential or partitioned data set) 201
QISAM 136, 141
QSAM 138—139
OUTPUT operand (OPEN macro)
described 120
overflow
area
independent 77
channel 134
exit address (PRTOV macro) 134
printer carriage 134
track
BDAM 40
overflow, track
BPAM 53
BSAM 70
QSAM 96
restrictions
chained scheduling 53, 96
OPTCD operand 96
overlay frame 161
overprinting 134

P

parallel data access block
See PDAB
parameter list construction
BUILDRCD macro 18
CLOSE macro 26
OPEN macro 125
READ macro 150
SETPRT macro 173—174
WRITE macro 206
parameter list modification
BUILDRCD macro 19
CLOSE macro 28
OPEN macro 127
READ macro 151
SETPRT macro 175—177
WRITE macro 207
partitioned data set
FIND macro 102
macro instructions used with 211
relationship to
BLDL macro 9—11
STOW macro 178—179
PDAB macro
use by access method 211
using 129
PDAB (parallel data access block)
constructing 129
generating a DSECT 130
symbolic field names 245
PDABD macro
symbolic field names 245
use by access method 211
POINT macro
described 131—132
MACRF operand
BSAM 67
NOTE macro 117
restriction
BSP macro 12
return codes 133
specified in MACRF operand
BPAM 52
use by access method 211
position feedback
current block 142, 197
next block 143, 203
positioning volumes
CHECK macro 20
CLOSE macro 23—25
FEOV macro 101
OPEN macro 119
POINT macro 131—132

position, relative key (RKP) 79
prefix, block
  See also BUFOFF operand
  effect on block length 57, 83
  effect on buffer length 83
  effect on data alignment 56, 81
print option for 3525
  BSAM 63, 65
  QSAM 87, 89
Print Services Facility
  See PSF
printer
  carriage control 30—135
  character set buffer loading 164
  control characters 223—224
  control information (SETPRT macro) 156
  control tape 134—135
  forms alignment 156
  forms control buffer loading 160
  skipping 30—32, 223—224
  spacing 30—224
program, channel
  BISAM 45
  BPAM 52
  BSAM 67
protection option, data
  BSAM 63, 65
  QSAM 88, 89
PRTOV macro
  described 134
  use by access method 211
PRTSP operand
  DCB macro
    BSAM 61
    QSAM 86
PSF (Print Services Facility)
  SYNAD routine 71
  SYS1.FDEFLIB 156
  SYS1.FONTLIB 156
  SYS1.PDEFLIB 156
PSPEED parameter
  described 163
  execute form 177
  list form 174
punch, card 62, 86
PUT macro
  data mode
    QSAM 92, 139
  locate mode
    QISAM 137
    QSAM 139
  move mode
    QISAM 136
    QSAM 139
  QISAM 136
  QSAM 119—121
  relationship with
    PRTOV macro 134
    SYNADAF macro 182
PUT macro *(continued)*
  relationship with *(continued)*
    TRUNC macro 193
  specified in DCB macro
    QISAM 76
    QSAM 92
  use by access method 211
PUTX macro
  described 141
  output mode 141
  specified in DCB macro
    QISAM 77
    QSAM 93
  TRUNC macro 193
  update mode 141
  use by access method 211

Q

QISAM (queued indexed sequential access method)
  general description 72
  macro instructions used with 211
  symbolic field names in DCB 238—242
QSAM (queued sequential access method)
  general description 80
  macro instructions used with 211
  symbolic field names in DCB 227—237

R

RDBACK operand (OPEN macro)
  restriction 120
read backward
  magnetic tape 120, 147
read column eliminate mode
  BSAM 64
  QSAM 87, 88
READ macro
  BDAM 142—144, 149
  BFTEK operand 56
  BISAM 144
  BPAM 147—148
  BSAM 147—149
  EODAD operand 51, 65
  execute form 151
  FIND macro 102
  FREEDBUF macro 105
  list form 150
  MACRF operand 52, 66—67
  NCP operand 52, 67
  relationship to
    BFTEK operand 34
    BUFL operand 35
    CHECK macro 20
    KEYLEN operand 37
    LIMCT operand 37
    MACRF operand 38, 44
    NCP operand 45
    OPTCD operand 39
    POINT macro 131

READ macro *(continued)*
relationship to *(continued)*
RELEX macro 152
SYNCDEV macro 188
WAIT macro 194
WRITE macro 196—198
specified in DCB macro
BDAM 38
BISAM 44
BPAM 52
BSAM 66
standard form
BDAM 142—144
BISAM 144
BPAM 147—148
BSAM (read direct data set) 149
BSAM (read sequential data set) 147—148
use by access method 211
reason codes
BLDL macro 11
BSP macro 13
FIND macro 103
SETPRT macro 170
STOW macro 180—181
RECFM operand
DCB macro
BDAM 40
BPAM 53
BSAM 70—71
QISAM 78
QSAM 95—97
record
area
construction 147
deletion option (ISAM) 77
format (see RECFM operand) 1
length (see LRECL operand) 1
descriptor word
BSAM 59
QISAM 73, 76
QSAM 91
physical
See BLKSIZE operand
retrieval 107—110, 142—149
segment 138
variable-length, spanned 82
writing 136—141, 196—204
recording density
magnetic tape
BSAM 60
QSAM 85
recording technique
magnetic
BSAM 60
magnetic tape
BSAM 60
QSAM 85
recovery procedure
tape error 69
register
contents on entry to
DCB exit routine 222
overflow exit routine 134
DCBD base 98—99
usage rules 6
register contents
exit list 222
relative addressing
BDAM 39
FIND macro 102
POINT macro 132
relative key position 79
release
buffer 104
buffer pool 106
dynamically acquired buffer 105
exclusive control 197
QSAM buffer 153
RELEX macro
described 152
MACRF operand 39
return codes 152
use by access method 211
relexp defined 6
relocatable expression defined 6
RELSE macro
use by access method 211
using 153
reorganization statistics (ISAM) 78
REREAD option
CLOSE macro 23
OPEN macro 121
restore data control block 23—25
return codes
BLDL macro 11
BSP macro 13
CLOSE macro 29
FIND macro 102
MSGDISP macro 116
NOTE macro 118
OPEN macro 123
POINT macro 133
RELEX macro 152
SETPRT macro 164—169
STOW macro 180—181
SYNADAF macro 184
SYNADRLS macro 187
WRITE macro 205
RETURN macro
SYNAD operand
BDAM 40
BISAM 47
BPAM 54
BSAM 71
QISAM 79, 225
QSAM 97

REWIND option
CLOSE macro 24
FEOV macro 101
REXMIT operand
SETPRT macro 164
RKP (relative key position) operand
DCB macro 76
record format information 79
R0 record
See capacity record

S

save area
general register requirements 6
SYNADAF requirement 182
SYNADRLS macro 187
scan mode
QISAM 76
search
BLDL macro
PDS directory 9—11
partitioned data set directory
FIND macro 102
type of
BDAM 38
QISAM 77
search argument
BDAM 38
QISAM 76
search direct option 95
search option, extended 39
segment
buffer 136
descriptor word 57, 149
interface, restriction 16
work area 36
sequential access methods
See access methods
services, optional
BDAM 39
BPAM 53
BSAM 68
QISAM 77
QSAM 93
SETL macro
described 154—155
ESETL macro 100
GET macro 107
use by access method 211
SETPRT macro
blocking/unblocking data checks 156
bypassing automatic forms positioning 157
execute form 175—177
list form 173—174
printing by print train or band 156
reason codes for 3800 170
return codes 164—169
selecting UCS and FCB images 156
SETPRT macro *(continued)*
standard form 156—172
use by access method 211
4248 printer
activation 157
deactivation 157
positioning 157
simple buffering 81
skipping, printer
See also spacing, printer
CNTRL macro 30
control characters 223—224
SMSI operand
DCB macro 46
SMSW operand
DCB macro 46
space, magnetic tape
backward 12, 30
forward 30
spacing, printer
See also skipping, printer
CNTRL macro 30
control characters 223—224
specified in DCB macro
BSAM 61
QSAM 86
spanned records
See variable-length, spanned records
STACK operand
DCB macro
BSAM 62, 64
QSAM 87, 88
stacker selection
CNTRL macro 30—32
control characters 223—224
DCB macro
BSAM 64
specified in DCB macro
BSAM 62
QSAM 87, 88
standard blocks
restriction with OPTCD operand 96
specifying 70, 96
statistics reorganization (ISAM) 78
status
following an I/O operation 209
STOW macro
described 178—181
directory action 179
reason codes 180
return codes 180—181
use by access method 211
suppress length indication (SLI)
relationship with channel programs 159
switching volumes
CHECK macro 20
FEOV macro 101

symbol defined 5
SYNAD exit routine
DCB macro
BSAM 71
GET macro 107, 110
relationship with
CHECK macro 20
CNTRL macro 32
DCB macro (see SYNAD operand) 1
POINT macro 132
PUT macro 137, 140
PUTX macro 141
SETL macro 155
SYNADAF macro 183
specifying in DCB macro
BDAM 40
BISAM 46
BPAM 54
QISAM 79
QSAM 96
SYNAD operand
DCB macro
BDAM 40
BISAM 46
BPAM 54
BSAM 71
QISAM 79
QSAM 96
SYNAD routine
PSF (Print Services Facility) 71
SYNADAF macro
described 182—183
message format 185
relationship with SYNADRLS macro 187
return codes 184
use by access method 211
SYNADRLS macro
described 187
relationship with SYNADAF macro 182
return codes 187
use by access method 211
SYNCDEV macro
data synchronization 188
synchronization of data to
tape 188
synchronizing I/O operations 20, 194—195
synchronous error exit
See SYNAD operand
SYSIN DD statement
BSP macro 12
CNTRL macro 30
FEOV macro 101
MACRF operand 67
NOTE macro 118
OPEN macro 119, 120
OPTCD operand 94
PUTX macro 141
RECFM operand 96
SYSIN DD statement *(continued)*
RELSE macro 153
SYSOUT DD statement
BSP macro 12
CNTRL macro 30
FEOV macro 101
MACRF operand 67
NOTE macro 118
OPEN macro 119, 120
OPTCD operand 94
POINT macro 132
PUTX macro 141
system-determined block size
BPAM 49
BSAM 57
QSAM 83

T

table reference character
See TRC
tape density
magnetic
BSAM 60
QSAM 85
tape error recovery procedure
BSAM 69
QSAM 94
tape recording technique
QSAM 85
temporary close
of data set 23
termination, abnormal
check routine 20
end-of-data
See EODAD operand
uncorrectable I/O error
See SYNAD operand
testing
completion of I/O 20, 194—195
for open data set 121—123
totaling exit
BSAM 66
list format 221
QSAM 91
track addressing, relative
BDAM 39
FIND macro 102
POINT macro 132
track index write, full 78
track overflow
BDAM 40
BPAM 53
BSAM 70
QSAM 96
translation
ASCII to EBCDIC
CHECK macro 20
GET macro 108
XLATE macro 208

translation *(continued)*
EBCDIC to ASCII
PUT macro 138
WRITE macro 201
XLATE macro 208
paper tape code 61
transmittal modes
See also MACRF operand
data 92, 110, 139
locate 107, 109, 139
move 107, 109, 136, 139
specifying 76
TRC (table reference character 3800) 68, 94, 162
TRTCH operand
DCB macro
BSAM 61
QSAM 85
TRUNC macro
described 193
specified in QSAM DCB 93
use by access method 211
truncating a block 193
TTR (track record address)
as used by BLDL macro 9
as used by FIND macro 102
as used by NOTE macro 117
TYPE=P (GET macro) 110
TYPE=T (CLOSE macro) 23—25

U

U-format records
BDAM 40
BPAM 54
BSAM 70
QSAM 96
UCS operand (SETPRT macro) 164
UCS (universal character set)
unblocking data checks 94
unblocking data checks
QSAM 94
SETPRT macro 163
uncorrectable I/O errors
See SYNAD operand
undefined length records
See U-format records
universal character set
See UCS
unmovable data sets
See DSORG operand
UPDAT operand
OPEN macro 120, 131, 147
restriction with POINT macro 131
restriction with READ macro 147
updating
partitioned data set directory 178—179
user
BLDL macro
data in PDS directory 9—11
user *(continued)*
data in partitioned data set directory
STOW macro 178—179
label exit
BSAM 66
list format 221
QSAM 91
totaling exit
BSAM 66
list format 221
QSAM 91
USING statement requirement
DCBD macro 98—99
PDABD macro 130

V

V-format records
BDAM 40
BPAM 54
BSAM 71
GET macro 108
QISAM 78
QSAM 96
validity checking
BDAM 39
BPAM 53
QISAM 78
QSAM 94
variable-length record (format-V)
See V-format records
variable-length, spanned records
See also V-format records
BFTEK 34, 56, 82
FEOV macro 101
PUT macro 138
restriction with
OPTCD operand 96
writing for BDAM 203
volume
forcing end 101
volume positioning
CHECK macro 20
CLOSE macro 23—25
FEOV macro 101
OPEN macro 119
POINT macro 131—132
volume switching
described 20
FEOV macro 101
VSAM (virtual storage access method)
referenced publications 1

W

WAIT macro
described 194—195
relationship to
CHECK macro 20
MACRF operand 38

WAIT macro *(continued)*
relationship to *(continued)*
READ macro 142, 145
WRITE macro 196, 199
use by access method 211
work area
BISAM
address of 47
size of 46
WRITE macro
execute form 207
list form 206
MACRF operand 52, 67
NCP operand 52, 67
relationship to
BUFL operand 35
CHECK macro 20
KEYLEN operand 37
LIMCT operand 37—38
MACRF operand 38, 44
NCP operand 45
OPTCD operand 39
POINT macro 131
PRTOV macro 134
READ macro 143, 145, 147
RELEX macro 152
SYNADAF macro 182
SYNCDEV macro 188
WAIT macro 194
return codes 205
specified in DCB macro
BDAM 38—40
BISAM 44
BPAM 52
BSAM 66—67
standard form
BDAM (create with BSAM) 203—205
BDAM (existing) 196—198
BISAM 199—200
BPAM 201
BSAM 201
testing for completion 20, 194—195
use by access method 211
WTOR macro
relationship with SETPRT macro 160

X

XCTL macro
relationship with BLDL macro 9
XLATE macro
use by access method 211
using 208
XLRI (extended logical record interface)
BUILDRCD macro 16
GET macro
use
QSAM 92

Numerics

31-bit addressing mode
executing macros 1
3262 Model 5 printer
COPYP parameter 159
maximum record length specification 213
3430 magnetic tape unit
record length 213
3480 magnetic tape subsystem
record length 213
3800 Model 3 printer 71, 156
4245 printer
maximum record length specification 213
4248 printer
COPYP parameter 159
maximum record length specification 213
SETPRT macro 157

MVS/ESA
Data Administration:
Macro Instruction Reference

SC26-4506-1

Reader's Comment Form

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 have applied any technical newsletters (TNLs) to this book, please list them here: _____

Comments (please include specific chapter and page references) :

Note: Staples can cause problems with automatic mail-sorting equipment.
Please use pressure-sensitive or other gummed tape to seal this form.

If you want a reply, please complete the following information:

Name _____ Date _____

Company _____ Phone No. (_____) _____

Address _____

_____

Thank you for your cooperation. No postage is 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 them directly to the address in the Edition Notice on the back of the title page.)

SC26-4506-1

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 MAIL PERMIT NO. 40 ARMONK, NY

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Department J57
P.O. Box 49023
San Jose, CA 95161-9945

Fold and tape | Please do not staple | Fold and tape

IBM

MVS/ESA
Data Administration:
Macro Instruction Reference

Reader's Comment Form

SC26-4506-1

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 have applied any technical newsletters (TNLs) to this book, please list them here: _____

Comments (please include specific chapter and page references) :

Note: Staples can cause problems with automatic mail-sorting equipment.
Please use pressure-sensitive or other gummed tape to seal this form.

If you want a reply, please complete the following information:

Name _____ Date _____

Company _____ Phone No. (_____) _____

Address _____

_____

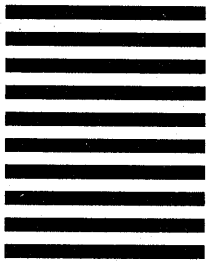
Thank you for your cooperation. No postage is necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail them directly to the address in the Edition Notice on the back of the title page.)

SC26-4506-1

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 MAIL PERMIT NO. 40 ARMONK, NY

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Department J57
P.O. Box 49023
San Jose, CA 95161-9945

Fold and tape | Please do not staple | Fold and tape

IBM®

IBM®

Program Number
5665-XA3

File Number
S370-34

SC26-4506-1

Printed in U.S.A.